MASTER EN SCIENCES INFORMATIQUES

Implémentation d'un langage interactif de programmation linéaire multicritère

Borlon, Jean-Luc

Award date:
1977

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Année Académique 1976-1977

Implémentation d'un langage interactif de programmation linéaire multicritère

JEAN-LUC BORLON

Mémoire
présenté en vue de l'obtention
du grade de
Licencié et Maître en Informatique

Je tiens à exprimer ma profonde gratitude à Monsieur J. FICHEFET pour l'intérêt porté à ce mémoire et pour ses conseils très précieux.

Qu'il me soit permis de remercier Monsieur J. GALLARD, responsable du département Management Science de la Compagnie CII-Honeywell-Bull à Paris, pour m'avoir permis de réaliser le stage qui donne son sens à ce travail.

Je suis également reconnaissant envers tous les membres de l'équipe pour leur accueil chaleureux et particulièrement envers Monsieur HUYNH THANH TAM dont la connaissance du MPS/66 m'a aidé pour la réalisation de l'interface.

TABLE DES MATIERES

°
° °

INTRODUCTION

Dans une démarche d'aide à la décision, on peut distinguer trois étapes :

1. L'analyse du problème qui permet d'éclairer l'objet des décisions et d'identifier les conséquences qui serviront à guider le choix.
2. La modélisation des préférences du décideur, relatives aux décisions possibles.
3. Le travail de résolution et d'interprétation des résultats.

En vue de permettre une modélisation plus proche de la réalité, ce travail voudrait fournir un outil de résolution pour un certain type de modèles.

Nous nous plaçons dans un contexte de programmation linéaire. Des techniques de résolutions largement éprouvées existent dans ce domaine. Quelques extensions ont déjà été proposées et réalisées : soit pour augmenter l'efficacité de la résolution (plus grand nombre de variables, précision des calculs, ...) c'est le cas des techniques de décomposition; soit pour mieux appréhender la réalité, non nécessairement linéaire, avec par exemple, la programmation en nombres entiers, ou la programmation séparable.

Dans ce travail, nous allons nous diriger vers une autre extension qui rejoint le deuxième point cité ci-dessus. Plutôt que de formaliser les préférences en un seul critère, il est intéressant de faire intervenir tous les décideurs pour l'élaboration de la solution finale. Quand nous mentionnons "les décideurs", il s'agit tout aussi bien de plusieurs responsables ayant chacun une fonction d'évaluation particulière, que de la personne confrontée à un problème qui possède plusieurs dimensions d'appréciation.

La première partie de ce travail va préciser le type de modèle qui a été retenu, et donner un algorithme de résolution après un rapide tour d'horizon des solutions existantes.

Dans la deuxième partie, nous examinons les spécifications générales d'un software de programmation linéaire. Cette étude est basée sur le MPS/66 (Mathematical Programming System) disponible par la série 60, niveau 66, du constructeur Honeywell Bull. Le but est d'évaluer les possibilités offertes pour l'implémentation d'un interface interactif reprenant l'algorithme défini dans la première partie.

La troisième partie détaille l'implémentation que nous avons réalisée sur le MPS/66. Elle comprend les spécifications du langage de commande qui est proposé, ainsi que la logique et les fonctions qu'il met en oeuvre. Quelques exemples de son fonctionnement sont donnés en annexe.

°

° °

PREMIERE PARTIE :

ETUDE DE L'ALGORITHME

INTRODUCTION

La programmation linéaire connaît beaucoup d'applications. Le principe de base consiste à trouver l'optimum d'une fonction linéaire quand les variables sont tenues de vérifier un système d'équations et/ou d'inéquations linéaires. Prenons comme exemple la constitution d'un alliage : les variables représentent les quantités des différents constituants, la fonction économique reprend le coût de l'alliage (somme des coûts unitaires multipliés par les quantités) et les contraintes de fabrication de l'alliage sont formalisées dans le système linéaire. Cependant, ce modèle a quelques limites. Celle qui nous préoccupe est le critère unique. Reprenons l'exemple : l'optimum minimisait les coûts de production. Mais ce n'est pas la seule politique; on peut avoir intérêt à minimiser les quantités de produits stockés pour diminuer l'encombrement des entrepôts. La solution optimale correspondante est vraisemblablement différente de la première. La notion d'optimum global doit donc être remplacée par celle de compromis : c'est la combinaison des variables qui réalisera l'équilibre entre tous les objectifs. Nous sommes ainsi passés à un problème multicritère.

Cet exemple élémentaire indique le type général de problème abordé. Face à une décision à prendre sur un ensemble d'alternatives possibles, plusieurs préférences, qui peuvent être conflictuelles vont orienter le choix. Différents types de modèles multicritères sont proposés par B. Roy [14].

o

o o

CHAPITRE I : LES ÉQUATIONS DU MODÈLE

Nous allons formaliser les notions intuitives qui ont été introduites et préciser les notations qui seront utilisées dans la suite du mémoire.

1. NOTATIONS

Soit le vecteur $x \in \mathbb{R}^n$ dont les composantes sont notées x^i, $i = 1, 2, \ldots, n$.
Il s'agit du vecteur colonne $(x^1, x^2, \ldots, x^n)'$ où ' indique la transposition.

Soit une matrice A, de dimension m x n

- a^i_j est l'élément de la ligne i et de la colonne j
- a_j représente la colonne j de A
- a^i représente la ligne i de A

2. LES EQUATIONS

2.1. Les variables

Soit $x = (x^1, x^2, \ldots, x^n)'$, le vecteur colonne appartenant à $\mathbb{R}^n$, des variables du problème.

2.2. Les objectifs

Supposons qu'il y ait r décideurs dont les préférences sont formalisées par les applications linéaires suivantes :

$$f^i : \mathbb{R}^n \longrightarrow \mathbb{R} : x \longrightarrow f^i(x) = \sum_{j=1}^{n} c^i_j x^j \qquad i = 1, 2, \ldots, r$$

Nous supposons également qu'il existe plus de deux fonctions linéairement indépendantes. En effet, le problème général, dans le cas de deux

fonctions seulement, se ramène à une paramétrisation :

$$\min_{x \in C} \lambda f^1(x) + (1-\lambda)\, f^2(x)$$

avec $0 \leqslant \lambda \leqslant 1$

C est l'ensemble engendré par les contraintes, il sera défini dans le paragraphe suivant.

Dans la suite du travail, on utilisera indistinctement les termes "fonction linéaire" et "application linéaire". De plus, vu le contexte général où nous nous sommes placés, "linéaire" sera souvent sous-entendu. Ces fonctions sont à minimiser. Il suffit de prendre l'opposé de la fonction pour passer d'une maximisation à une minimisation : $\max f(x) = - \min (-f(x))$. Dans la troisième partie, on détaillera ce processus de changement d'échelle.

Ces r fonctions définissent une application linéaire
$f : \mathbb{R}^n \longrightarrow \mathbb{R}^r : x \longmapsto f(x) = (f^1(x), f^2(x), \ldots, f^r(x))'$.

2.3. Les contraintes du problème

Elles sont reprises par $\sum_{j=1}^{n} a_j^i x^j \leqslant b^i \qquad i = 1, 2, \ldots, m$

$$x^j \geqslant 0 \qquad j = 1, 2, \ldots, n$$

Les a_j^i définissent une matrice A de dimension m x n. Si des contraintes sont exprimées par une autre relation que l'inégalité reprise ci-dessus, les modifications pour s'y ramener sont évidentes; pour une discussion détaillée, voir Simonnard [15], p. 9.

Les contraintes et les conditions de non négativité définissent un sous-ensemble de $\mathbb{R}^n$, qui sera noté C.

2.4. Le point utopique

Si $\hat{f}^i = \min_{x \in C} f^i(x) \quad i = 1, 2, \ldots, r$, alors $\hat{f} = (\hat{f}^1, \hat{f}^2, \ldots, \hat{f}^r)'$

est appelé point utopique du problème. Comme il est composé de l'optimum de chacune des fonctions, il représenterait l'optimum global du problème. Mais ce point de $\mathbb{R}^r$ n'est généralement l'image d'aucun point de C par l'application f.

o

o o

CHAPITRE II : INTERPRÉTATION GÉOMÉTRIQUE

Pour des questions de clarté, les représentations graphiques seront limitées à $\mathbb{R}^2$

1. ESPACE DES ACTIVITES

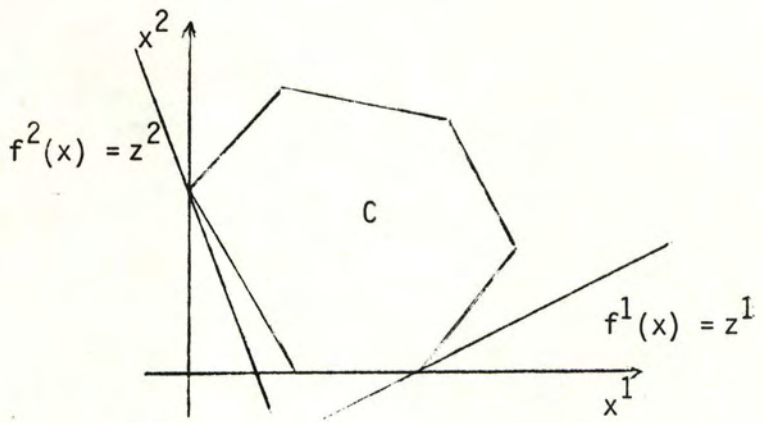
1.1. Propriétés de C

$\mathbb{R}^n$ est l'espace des activités, comme contenant les vecteurs x des variables du problème.

Dans $\mathbb{R}^n$:

- C est convexe (*) comme intersection de convexes
- on suppose C borné; C est donc un polyèdre convexe [15] , p. 253
- C est donc compact : ceci nous permet d'affirmer que toute fonction continue définie sur C y atteint ses bornes; en particulier, les objectifs sont des fonctions continues.

1.2. Figure



(*) On appelle "ensemble convexe C" un ensemble de points tel que

si $x_1, x_2 \in C$ et $0 \leq \lambda \leq 1$ alors $\lambda x_1 + (1-\lambda)\, x_2 \in C$

Dans $\mathbb{R}^n$, une droite, un hyperplan, un demi-espace sont des ensembles convexes

2. ESPACE DES OBJECTIFS

2.1. L'image de C par l'application linéaire f

C'est $\mathbb{R}^r$ qui contient les vecteurs dont chacune des composantes est la valeur d'un des objectifs du problème.

f(C), image de C par l'application linéaire f est une partie compacte (l'image continue d'un compact est compacte, voir [5] p. 64)

De plus, f(C) est également convexe comme image d'un convexe par une application linéaire.

Démonstration

il faut montrer que :

$$\forall\, g,h \in f(C) : \lambda\, g + (1-\lambda)\, h \in f(C) \qquad , 0 \leqslant \lambda \leqslant 1$$

Or, par définition de f(C)

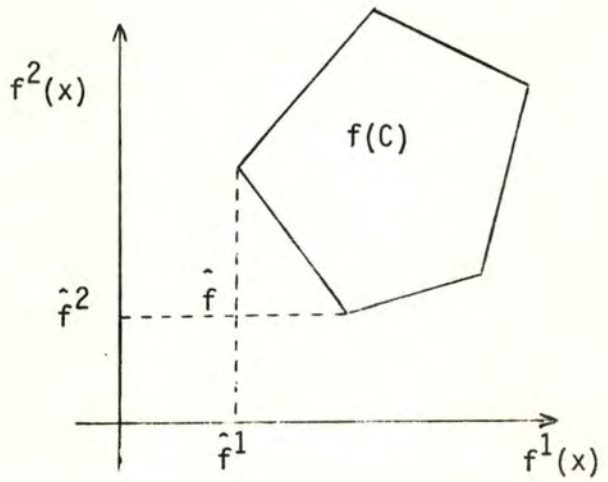
$$\exists\, x_1 \in C \text{ tel que } f(x_1) = g,$$
$$\exists\, x_2 \in C \text{ tel que } f(x_2) = h,$$

d'où :

$$\begin{aligned} \lambda g + (1-\lambda)\, h &= \lambda f(x_1) + (1-\lambda)\, f(x_2) \\ &= f(\lambda\, x_1 + (1-\lambda)\, x_2) && \text{car f est linéaire} \\ &\in f(C) && \text{car C est convexe} \end{aligned}$$

C'est dans l'espace des objectifs que se fera la recherche du meilleur compromis puisqu'on y a accès aux valeurs des fonctions et que l'on fait l'hypothèse que les décideurs évaluent une solution par les valeurs que prennent les objectifs.

2.2. Figure



généralement $\hat{f}$, le point utopique, n'est l'image d'aucun point de C.

CHAPITRE III : EVALUATION D'UNE SOLUTION

Il a déjà été dit que si $\hat{f} \in f(C)$, le problème serait résolu.

Dans le cas contraire, le concept de solution optimale doit être remplacé par celui de "meilleur" compromis entre les objectifs.
Nous appellerons compromis, un élément de $f(C)$.

P.L. Yu a présenté un cadre théorique général pour le concept de "meilleur" compromis. Le but de ce mémoire n'étant pas d'étudier les aspects théoriques de la programmation multicritère, nous nous limiterons à donner quelques résultats, en insistant sur l'interprétation qu'ils recouvrent, ainsi que sur un cas particulier, une solution optimale au sens de Pareto, que nous affinerons encore par la suite.

1. STRUCTURES DE DOMINANCE ET SOLUTIONS NON DOMINEES

1.1. Structures de dominance

Soient g et $h \in f(C)$, deux compromis qui sont soumis à l'appréciation des décideurs.

Si g est préféré à h, et s'il existe un vecteur $d \in \mathbb{R}^r$ tel que $h = g + \lambda d$, avec $\lambda > 0$, alors d est appelé un facteur de dominance de g.

On va noter $D(g)$, l'ensemble de tous les facteurs de dominance de g, plus le vecteur nul de $\mathbb{R}^r$.

$\{D(g) \mid g \in f(C)\}$ est la structure de dominance de notre problème de décision.

1.2. Solutions non dominées

h est dominé par g ssi $h \in g + D(g)$

Cette dernière expression n'est qu'une convention d'écriture pour $\exists d \in D(g) : h = g + d$

h est une solution non dominée ssi

$\nexists g \in f(C)$, $g \neq h$ tel que $h \in g + D(g)$

ce qui signifie qu'il n'est pas possible de trouver une solution meilleure que h, dans le cadre des préférences que l'on a choisies, et qui sont formalisées par la structure de dominance $\{D(g) \mid g \in f(C)\}$

1.3. Synthèse

D'un point de vue théorique, le problème est ainsi posé

- se donner une structure de dominance
- rechercher des points de f(C) et qui sont non-dominés.

2. EXEMPLES

2.1. Solution optimale au sens de Pareto

En premier lieu, nous allons considérer le cas particulier qui sera pris comme référence pour le reste du travail. Ce choix sera justifié plus loin.

Soit $D(g) = D_0 = \{d \in \mathbb{R}^r \mid d^i \geqslant 0, \quad i = 1, 2, \ldots, r\}$

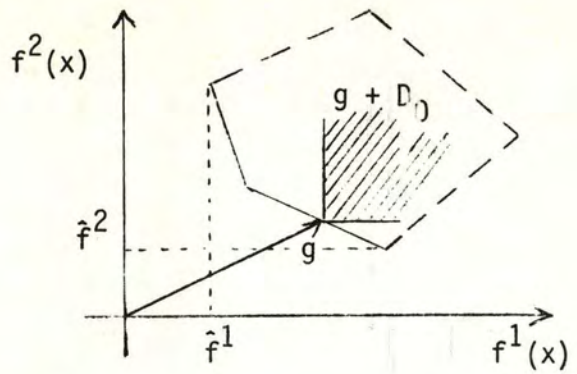
Une solution non dominée pour cette structure de dominance correspond à une solution optimale au sens de Pareto :

$x \in C$ est une solution optimale au sens de Pareto (ou solution efficiente, ou efficace) ssi

$$\nexists y \in C \text{ et } y \neq x \; : \; f^i(y) \leqslant f^i(x) \qquad \forall i = 1, 2, \ldots, r$$
$$f^i(y) < f^i(x) \qquad \text{pour au moins un } i$$

Cela correspond au fait que l'on a un "bon" compromis quand il n'est pas possible de l'améliorer sur chacune des composantes.

Figure :



les faces en traits plains du polyèdre f(C) correspondent aux solutions efficientes.

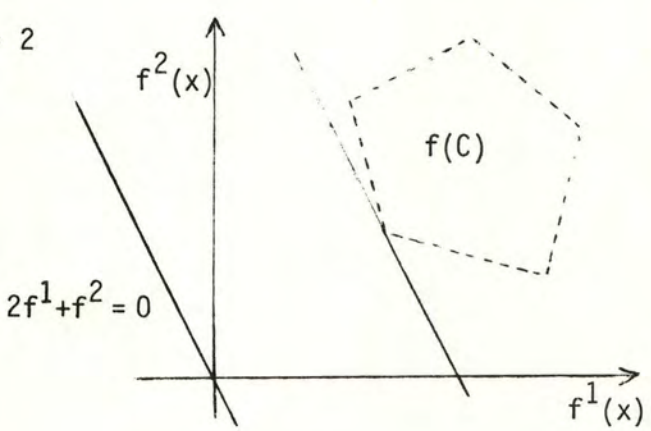
Cette structure de dominance correspond à une partie de $\mathbb{R}^r$: $1/2^r$ de l'espace complet : "quadrant" de l'espace où toutes les coordonnées sont positives ou nulles. Cette structure ne fait en aucune manière présupposition d'une quelconque relation de préférence ou de hiérarchisation entre les objectifs.

2.2. Pondération des objectifs

A l'opposé, il existe le cas où les décideurs peuvent donner exactement l'importance relative de chacun des objectifs : le problème à plusieurs critères se ramène à une optimisation de la combinaison linéaire des objectifs.

Soit $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$ avec $\lambda_i > 0, \quad \forall i = 1, 2, \ldots, r$ le vecteur des coefficients représentant l'importance relative des différents objectifs. Le problème revient à minimiser $\sum_{k=1}^{r} \lambda_k f^k(x)$ avec $x \in C$.
La structure de dominance qui intervient alors est $D_\lambda = \{d \in \mathbb{R}^r \mid \lambda \cdot d \geq 0\}$
Elle correspond à un demi-espace de $\mathbb{R}^r$, engendré par l'hyperplan défini par le vecteur λ.

Figure : si $\frac{\lambda_1}{\lambda_2} = 2$



La solution non dominée est dans ce cas unique; en effet, on est ramené à de la programmation linéaire dans l'espace des objectifs et la solution est unique sauf si une arête de f(C) est parallèle à l'hyperplan défini par λ.

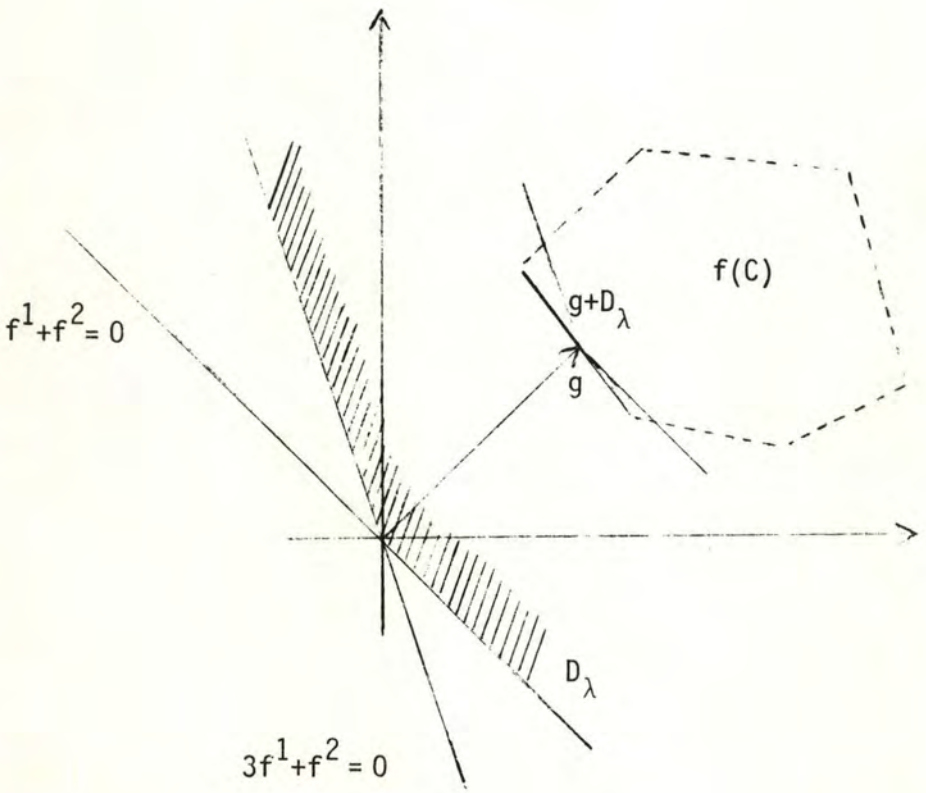
Les coefficients λ_i peuvent s'interpréter comme des coûts de substitution entre les objectifs; dans l'exemple, le décideur accepte de perdre 2 unités sur l'objectif 2 pour en gagner 1 sur l'objectif 1,et inversément.
Fixer λ implique une hypothèse très forte sur la connaissance que le décideur a sur ses objectifs.

2.3. Substitution entre les objectifs partiellement connue

On peut considérer une solution intermédiaire. Les décideurs ne connaissent plus avec précision les taux de substitution entre les objectifs, mais ils peuvent leur fixer des bornes. Supposons qu'ils évaluent le rapport λ_1/λ_2 par $1 < \lambda_1/\lambda_2 < 3$

$D_\lambda = \{d \in \mathbb{R}^2 \mid d^1 + d^2 \geqslant 0 \text{ et } 3d^1 + d^2 \geqslant 0\}$ voir [18] p. 255 pour la justification.

Figure



3. CONCLUSIONS

On constate que plus l'information sur les préférences entre les objectifs est grande, plus la structure de dominance est grande. Elle passe d'un quadrant dans le cas de l'optimalité au sens de Pareto à un demi-espace quand on fixe les coefficients de la combinaison linéaire. Ces propriétés qui sont présentées uniquement intuitivement sont démontrées dans [18].

YU souligne que la différence entre la structure D_λ présentée au paragraphe 2.3. et D_0 la structure pour l'optimalité au sens de Pareto, résulte des informations que les décideurs peuvent fournir sur leurs préférences et qui sont perdues par Pareto. Il est en effet vraisemblable qu'ils ont une connaissance a priori du problème qui leur permette d'évaluer des coûts de substitution entre les objectifs, au moins entre certaines bornes.
Cependant, d'autres auteurs tels que WALLENIUS [16] insistent sur la difficulté qui surgit quand on doit donner une évaluation quantitative des préférences entre les objectifs. DYER écrit également dans [7] p. 211, au sujet d'essais qu'il a effectués : "Subject 9 suggested that he would assign cost a weight of.1, horsepower a weight of .2, and mileage a weight of .6. However, subject 9's response to queries posed by V.M. algorithm indicated that his weights(at a peculiar operating point) were -1 for cost, 16 for horsepower, and 200 for mileage".

De plus, nous introduirons de manière beaucoup plus naturelle la connaissance a priori que les décideurs ont du problème. C'est pourquoi nous avons retenu la structure de dominance qui considère tous les objectifs comme indiscernables.

o

o o

CHAPITRE IV : PRÉSENTATION DE QUELQUES ALGORITHMES

Avant d'aller plus loin, quelles sont les caractéristiques qui permettent de choisir un type d'algorithme ?

Dans un contexte multi-critères, le concept de solution optimale a été remplacé par celui de "meilleur" compromis. Il s'agit en quelque sorte d'un équilibre à rechercher entre les différents décideurs. On pourrait donc tout d'abord exiger d'un algorithme qu'il ait un caractère interactif qui permettra de tenir compte des réactions des décideurs lors de l'élaboration de cet équilibre. Toujours dans le cadre d'un travail interactif, il est souhaitable que le dialogue soit assez souple : c'est-à-dire que les décideurs doivent pouvoir corriger ou ajuster des modifications qui ont été introduites auparavant et qui n'ont pas amené les résultats espérés.

D'autre part, J. FICHEFET [8] note à la suite de plusieurs auteurs que l'optimum d'un objectif n'est pas nécessairement la solution qui aura l'approbation du décideur; il préférera souvent un seuil de satisfaction qu'il s'est fixé. Le fait de considérer un modèle qui tient compte des niveaux de satisfaction a plusieurs avantages. La valeur qui est fixée pour cible découle de l'expérience passée de l'entreprise, et pose les buts à atteindre pour l'avenir, ce qui permet de garder une meilleure cohérence dans la prise de décision.
Nous avons ainsi introduit de manière beaucoup plus naturelle, la connaissance que les décideurs ont de leur entreprise : chaque responsable annonce un seuil qu'il souhaite atteindre dans son département, plutôt que de chercher à évaluer des taux de substitution par rapport aux autres objectifs.

Finalement, on peut encore citer quelques points qui retiendront l'attention lors de la présentation de chaque algorithme. Il y a évidemment la la simplicité de compréhension de la démarche. Ensuite, le nombre d'informations qui seront fournies par l'algorithme pour permettre aux décideurs de progresser dans la recherche du compromis, et inversément, la quantité d'informations que les décideurs doivent donner lors de chaque phase de dialogue. Une information très intéressante consiste en ce que nous appellerons, par référence à la littérature habituelle, "une matrice de paiement".

Elle regroupe l'optimum respectif de chacun des objectifs ainsi que la valeur des autres fonctions lors de la réalisation des différents optima.

Soient P cette matrice, et x_i tels que $f^i(x_i) = \hat{f}^i = \min_{x \in C} f^i(x)$

$i = 1, 2, \ldots, r$

(x_i est la solution qui réalise l'optimum i), alors

$$P^i_j = f^i(x_j)$$

On obtient $P^i_i = \hat{f}^i$: les valeurs optimales de chacune des fonctions se trouvent sur la diagonale de P. Cette matrice permet d'avoir une vue globale sur le problème. L. DEGEHET [4] en a présenté les avantages, tant au point de vue économique décisionnel.

1. COMBINAISON LINEAIRE DES OBJECTIFS

Le principe a déjà été donné plus haut : les décideurs doivent fournir un vecteur $\lambda=(\lambda_1, \lambda_2, \ldots, \lambda_r)$ dont les composantes représentent l'importance relative de chacun des objectifs. La solution est donnée par

$$\min_{x \in C} \sum_{i=1}^{r} \lambda_i . f^i(x)$$

Nous avons déjà insisté sur le côté généralement non réaliste de ce vecteur λ. Il faut aussi remarquer que l'on ignore les valeurs optimales que prennent les objectifs indépendamment. On pourrait cependant étudier le voisinnage du compromis obtenu en effectuant une paramétrisation de la fonction objectif.

Soit $\Delta\lambda$ la perturbation à introduire sur λ. Il faut étudier le comportement de la solution pour différentes valeurs du paramètre ψ dans

$$\min_{x \in C} \sum_{i=1}^{r} (\lambda_i + \psi . \Delta\lambda_i) f^i(x)$$

Mais il est à nouveau difficile de choisir le vecteur de perturbation $\Delta\lambda$, et si on le prend successivement égal aux r vecteurs(0 ...1 ...0) de $\mathbb{R}^r$, le procédé devient très lourd.

Ce type d'approche s'insère dans un contexte plus général. Il considère que les aspirations qui ont été exprimées dans les différents critères peuvent s'aggréger en une fonction d'utilité unique $\cup$ (f(x)). Cette fonction fait intervenir tellement de facteurs humains qu'elle est généralement très complexe. Plutôt que de forcer les décideurs à préciser leur fonction d'utilité unique, beaucoup d'algorithmes ne la font pas intervenir et travaillent sur tous les critères à la fois. Dans le cas particulier ci-dessus, elle est simplifiée puisqu'elle est supposée être une combinaison linéaire pondérée des objectifs.

2. METHODES D'ENUMERATION DES SOLUTIONS EFFICACES

Au chapitre III, nous avons vu qu'après avoir choisi une structure de dominance, il restait à fournir le compromis parmi les solutions non dominées. L'ensemble de toutes les solutions non dominées (dans l'espace des objectifs) est un sous-ensemble de f(C) qui possède des propriétés intéressantes : il est composé de faces adjacentes du polyèdre f(C). Cela a suggéré à des auteurs tels que YU [18], [19], [20] et ZELENY [21], une méthode de résolution.

1) Rechercher toutes les solutions extrémales non dominées; c'est-à-dire les solutions qui ne sont pas combinaison linéaire d'autres solutions. Cet ensemble est nécessairement fini puisqu'un polyèdre possède un nombre fini de faces. Dans le cas où la structure de dominance correspond aux solutions optimales au sens de Pareto, la démarche se ramène à une extension de l'algorithme du simplexe. Quand l'importance relative des objectifs est connue partiellement (§ 2.3, Chap. III), YU [18] a également proposé un algorithme.

2) Pouvoir engendrer toutes les solutions non dominées par combinaison linéaire des solutions extrémales. Plus exactement, le problème consiste à déterminer toutes les faces de f(C) qui sont non dominées; elles sont de nouveau en nombre fini.

Ce type de résolution ne paraît pas très adapté, étant donnés les objectifs qui ont été fixés. Bien qu'il permette de donner toutes les solutions efficaces, donc évidemment celle qui satisfera le plus les décideurs; il ne donne aucun moyen pour guider la recherche en éliminant certaines possibilités qui n'ont aucun intérêt dans la pratique. Il fournit trop de solutions et néglige l'aspect interactif nécessaire à l'élaboration du compromis. De plus, les décideurs n'ont pas connaissance de la matrice de paiement et la notion de cible n'est pas incluse dans la démarche.

Et finalement, on peut citer YU [20] p. 467 "Of course, to actually apply the results to practical decision problem remains to be a challenging task".

3. METHODE INTERACTIVE DE GEOFFRION, DYER et FEINBERG [6],[7],[9]

On suppose l'existence d'une fonction d'utilité $\cup (f(x))$ qu'il faut maximiser sous la contrainte $x \in C$; elle restera cependant implicite. L'algorithme emprunte une démarche de programmation non linéaire, mais il peut évidemment s'appliquer au cas linéaire.

On fait les hypothèses suivantes :

- C est un ensemble convexe et compact ;
- $\cup$ et les fonctions économiques (qui sont à maximiser), sont des fonctions concaves et continûment différentiables sur C ;
- $\cup$ est croissante en chaque f^i.

Partant d'un point x_k, choisir une direction qui fait croître l'utilité globale. Dans cette direction, résoudre un problème d'optimisation unidimensionnel pour trouver le point x_{k+1}.

Etape 0 : le décideur choisit un point initial $x_1 \in C$; $k := 1$

Etape 1 : - le décideur fournit les poids de substitution w_i^k
- calculer y_k tel que y_k soit solution de

$$\max_{y \in C} \sum_{i=1}^{r} w_i^k \cdot \nabla_x f^i(x_k).y$$

ce qui dans le cas linéaire est équivalent à

$$\max_{y \in C} \sum_{i=1}^{r} w_i^k . f^i(y)$$

- faire $d_k = y_k - x_k$

Etape 2 : - donner les valeurs $f^i(x_k+t.d_k)$ pour $0 \leqslant t \leqslant 1$ et laisser déterminer par le décideur la solution optimale t_k au problème

$$\max_{0 \leqslant t \leqslant 1} \cup [f^1(x_k+t.d_k), f^2(x_k+t.d_k), \ldots, f^r(x_k+t.d_k)]$$

ce qui consiste à choisir le meilleur vecteur $(f^1, f^2, \ldots, f^r)'$ parmi ceux qui viennent d'être tabulés.

- $x_{k+1} := x_k + t_k.d_k$; $k := k+1$; aller à l'étape 1.

les poids w_i^k

Soit le gradient de $\cup(f(x))$, $\nabla_x \cup(f(x_k)) = \nabla_f \cup(f(x_k)) \cdot f'(x_k)$ où $f'(x_k)$ est la matrice jacobienne de f au point x_k ; une ligne de cette matrice a été notée $\nabla_x f^i(x_k) = (\frac{\partial f^i}{\partial x^1}(x_k), \frac{\partial f^i}{\partial x^2}(x_k), \ldots, \frac{\partial f^i}{\partial x^n}(x_k))$.

Le problème revient à estimer le vecteur $\nabla_f \cup(f(x_k)) =$

$$(\frac{\partial \cup}{\partial f^1}(f(x_k)), \frac{\partial \cup}{\partial f^2}(f(x_k)), \ldots, \frac{\partial \cup}{\partial f^r}(f(x_k))).$$

Les auteurs ont proposé la solution suivante :

$\nabla_f \cup(f(x_k))$ est un vecteur colinéaire à $(1, \frac{\partial \cup/\partial f^2}{\partial \cup/\partial f^1}, \ldots, \frac{\partial \cup/\partial f^r}{\partial \cup/\partial f^1})$.

En outre, on peut approximer $\frac{\partial \cup/\partial f^i}{\partial \cup/\partial f^1}$ par $\frac{\Delta f^1}{\Delta f^i}$, $i = 2, 3, \ldots, r$, qui représentent les coûts de substitution de chacun des objectifs par rapport à f^1, arbitrairement pris comme référence.

Il faut noter que les coûts de substitution ne sont plus fixes comme dans le cas de la combinaison linéaire des objectifs, mais qu'ils dépendent du point x_k, la solution courante. Dyer [6] et [7] donne une procédure interactive permettant d'évaluer ces coûts. Elle est basée sur une série de comparaisons qualitatives. Le programme demande de choisir entre le point $A = (f^1(x_k), f^2(x_k), \ldots, f^r(x_k))$ et $B = (f^1(x_k) + \Delta f^1, f^2(x_k), \ldots, f^j(x_k) - \Delta f^j, \ldots, f^r(x_k))$ quand on veut évaluer w_j^k. Une modification de l'algorithme de bissection est appliquée, qui augmente Δf^j si on préfère B à A et qui diminue Δf^j dans le cas contraire. A l'indifférence, w_j^k est estimé par le rapport $\Delta f^1/\Delta f^j$.

La dernière partie de l'étape 1 peut s'interpréter par : on cherche à se déplacer dans le sens du gradient positif (pour faire croître $\cup(f(x))$) tout en vérifiant les contraintes du problème ($y \in C$).

Comme C est convexe et que $x_k \in C$ et $y_k \in C$, on peut affirmer que $x_{k+1} \in C$.

Le dialogue est établi avec l'utilisateur à deux étapes : pour l'évaluation des poids w_i^k ainsi que pour le choix de t_k. Ces étapes interactives sont donc constructives et permettent aux décideurs de formuler progressivement leurs souhaits. Cependant, ils n'ont pas la connaissance de la matrice de paiement. Pour ce qui est des niveaux de satisfaction, s'ils ne les fournissent pas explicitement, on peut supposer que c'est bien pour les atteindre que les décideurs orientent leurs modifications aux étapes 1 et 2. Le reproche majeur qui est souvent fait à cette méthode réside dans la fonction d'utilité $\cup(f(x))$. Nous avons déjà souligné sa complexité. Et si elle n'apparaît pas explicitement, les décideurs doivent néanmoins en fournir des indications pour l'évaluation des w_i^k. Notons que la routine basée sur les comparaisons qualitatives (qui n'est pas mentionnée dans [9])semble éliminer partiellement cette difficulté. En dernier lieu, il faut remarquer que rien ne permet d'affirmer que la solution qui est atteinte est efficace.

4. GOAL PROGRAMMING

Cette méthode non itérative a été introduite par Charnes et Cooper : [3]. Chaque décideur propose son niveau de satisfaction M^k. Les décideurs déterminent ainsi un vecteur $M \in \mathbb{R}^r$. Le problème consiste à trouver le point $x^\star$ tel que le vecteur $f(x^\star)$ soit "aussi proche que possible" du vecteur M. Ce qui se formule par exemple par : chercher $x^\star \in C$ qui réalise $\min_{x \in X} d_p(f(x),M)$

où $d_p(f(x),M)$ est la distance induite par le norme L_p dans $\mathbb{R}^r$

$$d_p(f(x),M) = \left(\sum_{i=1}^{r} |f^i(x) - M^i|^p \right)^{1/p} \quad \text{si } 1 \leqslant p < \infty$$

$$= \max_{i=1,2,\ldots,r} |f^i(x) - M^i| \quad \text{si } p = \infty$$

Deux cas peuvent se ramener à la résolution d'un programme linéaire:

1) $p = 1$

$$\min_{x \in C} d_1(f(x),M) \Leftrightarrow \min_{x \in C} \sum_{i=1}^{r} |f^i(x) - M^i|$$

$$\Leftrightarrow \min \sum_{i=1}^{r} (y^{+i} + y^{-i})$$

$$\text{s.c. } f^i(x) + y^{+i} - y^{-i} = M^i$$

$$x \in C,\ y^{+i} \geqslant 0,\ y^{-i} \geqslant 0 \quad i=1,2,\ldots,r$$

2) $p = \infty$

$$\min_{x \in C} d_\infty(f(x),M) \Leftrightarrow \min_{x \in C} \left(\max_{i=1,2,\ldots,r} |f^i(x) - M^i| \right)$$

$$\Leftrightarrow \min\ y$$

$$\text{s.c. } f^i(x) - M^i \leqslant y$$

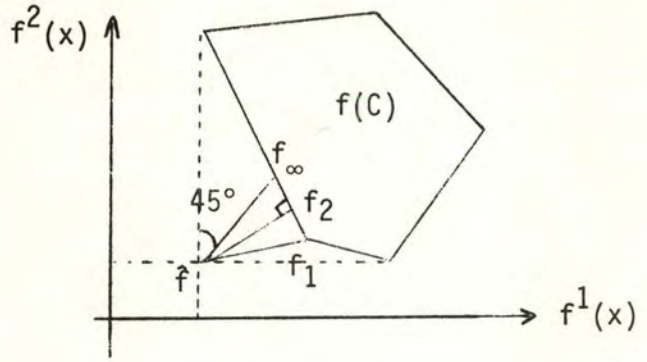
$$M^i - f^i(x) \leqslant y \quad i=1,2,\ldots,r$$

$$x \in C,\ y \geqslant 0$$

Quelle est l'influence du choix de la norme ? Bien qu'il soit démontré en analyse que les différentes normes sur $\mathbb{R}^n$ sont équivalentes, c'est-à-dire qu'elles ont des résultats identiques pour les problèmes de convergence, elles ne fournissent pas les mêmes résultats "graphiques". Il suffit pour s'en

convaincre de tracer les différentes boules de rayon unitaire et centrées à l'origine, correspondant à chacune des normes. Dans le cas qui nous occupe, le point qui réalise le minimum de la distance ne sera vraisemblablement pas le même. En effet, Yu [17] en a fait l'étude théorique. Ces développements dépassent le cadre de ce travail, nous allons simplement en donner les résultats qualitatifs.

figure



explication : on suppose pour la clarté de la figure que M, le vecteur des cibles est confondu avec $\hat{f}$.

f_p est le point qui réalise le minimum de la distance induite par L_p.

Les résultats de Yu sont : tous les points se trouvent sur la même arête du convexe f(C), dans l'intervalle $[f_1, f_\infty]$.

Ceci souligne l'influence que peut avoir sur le compromis obtenu le choix de l'une ou l'autre norme.

C'est le premier algorithme rencontré qui introduit le notion de niveau de satisfaction, ou cible. Il a cependant quelques limites. Les décideurs ne sont pas informés de la matrice de paiement ; ils ne peuvent donc faire de comparaisons entre la cible et l'optimum. Ils ont à choisir une distance (qui peut être en partie fixée par la technique de résolution disponible : distance L_1 ou L_∞ si on utilise la programmation linéaire). De plus, l'aspect interactif est éliminé puisque les décideurs n'ont qu'à introduire les cibles et que l'algorithme ne fournit pas directement de moyens pour étudier le voisinage du compromis obtenu.

5. METHODE STEM [9]

Présentation de la méthode :

Etape 0 : construire la matrice de paiement en optimisant successivement chacun des $f^i(x)$ $\quad i = 1, 2, \ldots, r$

Etape 1 : poser $C_1 = C$; $\quad k := 1$

calculer $\alpha^j = \left| \dfrac{p_j^j - m^j}{p_j^j} \right| \dfrac{1}{\sqrt{\sum_{i=1}^{n} (C_i^j)^2}}$ $\quad j = 1, 2, \ldots, r$

avec m^j = la plus grande valeur de la j^e ligne de la matrice P.

$$\Pi^j = \frac{\alpha^j}{\sum_{i=1}^{r} \alpha^i} \qquad j = 1, 2, \ldots, r.$$

Etape 2 ou étape de calcul :

résoudre $\quad \min \lambda$

$$\lambda \geqslant (f^j(x) - \hat{f}^j).\ \Pi^j \qquad j = 1, 2, \ldots, r$$
$$\lambda \geqslant 0,\ x \in C_k$$

la solution est x_k.

Etape 3 ou étape de décision :

. y a-t-il des composantes de $f(x_k)$ qui sont satisfaisantes ?
aucune : le problème n'a pas de solution
toutes : x_k est la solution demandée.

. soit $f^j(x_k)$ une composante satisfaisante : essayer de la relâcher pour améliorer les autres composantes.
$\Delta f^j > 0$ est la relaxation consentie sur l'objectif j.

$$C_{k+1} = C_k \cup \{f^j(x) \leqslant f^j(x_k) + \Delta f^j\}$$
$$\cup \{f^i(x) \leqslant f^i(x_k)\ ,\ i = 1, 2, \ldots, r \text{ et } i \neq j\}$$

$\Pi^j := 0$; $k := k+1$; aller à l'étape 2.

Les poids Π^j sont introduits pour des questions de normalisation. Leur justification sera détaillée au chapitre V.

Les principaux avantages sont les suivants :

- il ne demande aucune information a priori sur l'importance relative des objectifs ;
- les décideurs peuvent consulter la matrice de paiement ;
- c'est une méthode interactive qui permet aux décideurs de construire progressivement le compromis au cours de l'étape de décision.

Il y a cependant quelques problèmes :

- une fois qu'un objectif a subi une relaxation, il ne peut plus être modifié ; ni pour corriger la modification démandée, ni pour en proposer une nouvelle. Ce qui, par contre, assure la convergence en r-1 itérations.
- les décideurs n'ont pas la possibilité de préciser des seuils de satisfaction.

Un autre problème peut surgir, c'est le choix de l'objectif à relâcher ainsi que le montant de la modification. Les auteurs proposent en annexe de leur article une ébauche de réponse : il s'agit d'aider les décideurs par une étude du comportement de la solution pour des modifications unitaires de chacun des objectifs, en quelque sorte, une analyse de sensibilité. Nous détaillerons ce type d'aide aux décideurs dans le chapitre V.

6. AUTRES METHODES

La plupart des algorithmes de programmation linéaire multicritère son apparentés aux catégories qui ont déjà été mentionnées. On peut cependant citer un dernier type. Il s'agit de l'algorithme de Belenson et Kapur [1] dérivé de la théorie des jeux. C'est une méthode interactive où les décideurs indiquent uniquement l'objectif à améliorer. Les principaux désavantages sont que chaque objectif ne peut être choisi qu'une seule fois et que les décideurs ne contrôlent pas l'exploration des solutions efficaces puisque le montant de la modification est calculé automatiquement. Et finalement, l'algorithme ne tient pas compte des niveaux de satisfaction que les décideurs peuvent proposer.

7. SYNTHESE

Après ce rapide survol des algorithmes, ceux qui paraissent le plus prometteurs sont la méthode STEM et l'algorithme de Geoffrion, Dyer et Feinberg. Ils permettent en effet la meilleure élaboration interactive du compromis à fournir aux décideurs et, moyennant quelques adaptations à définir, il est possible de combler les faiblesses qui ont été mentionnées.

Cependant, d'autres critères vont affiner le choix. Un des buts de ce mémoire était d'arriver à la constitution d'un interface de programmation multicritère, interface à inclure dans un package de programmation linéaire existant de manière opérationnelle. Le principal avantage de cette démarche était que si l'interface était intelligemment intégré, l'ensemble formerait directement un outil pour résoudre des problèmes réels. La puissance des programmes-produits (taille du problème, précision des calculs, résultats intermédiaires, ...) pourrait être utilisée de manière intégrée pour la résolution de problèmes multicritères.

D'autre part, dans le souci d'avoir un interface utilisable, il est bon de rappeler les résultats d'une étude comparative entre différents algorithmes interactifs de programmation multicritère (Wallenius [16]). L'étude a porté sur trois algorithmes : STEM, Geoffrion - Dyer - Feinberg et une approche non structurée qui va être décrite. L'utilisateur fournit de manière subjective un vecteur f_1. Le programme vérifie que f_1 appartient à f(C). S'il en est ainsi, l'utilisateur peut introduire un nouveau vecteur pour améliorer le compromis précédent, sinon le programme indique les contraintes qui ne sont pas vérifiées, et la boucle reprend.

Plusieurs mesures de performance étaient considérées :

- la confiance que les décideurs ont dans le compromis final ;
- la facilité d'utilisation ;
- la facilité de compréhension de la méthode ;
- l'utilité des informations fournies pour aider l'utilisateur ;
- la rapidité de convergence, mesurée par le nombre d'itérations et le temps total d'exécution de l'algorithme ;
- le temps CPU.

Il faut noter que la méthode STEM était implémentée sans l'analyse de sensibilité proposée par les auteurs.

Les conclusions sont les suivantes :

- la performance globale de la méthode de Geoffrion a été beaucoup moins bonne que celle prévue par d'autres articles (Dyer [7]). L'obstacle principal étant l'évaluation des taux de substitution entre les objectifs, et cela malgré la routine qui calcule ces taux à partir de comparaisons qualitatives. Les différences par rapport à STEM sont surtout marquées dans la facilité d'utilisation, la facilité de compréhension de la démarche et le temps total nécessaire pour résoudre le problème.
- L'approche non structurée a obtenu des résultats très compétitifs par rapport aux deux autres techniques plus sophistiquées. Wallenius explique ce résultat par la taille non réaliste du problème utilisé pour les tests : il était beaucoup trop restreint. Face à un problème réel, il est plus difficile de pouvoir fournir une solution réalisable pour le vecteur f.

Toutes ces raisons contribuent à orienter le choix vers une démarche de type STEM après y avoir introduit des modifications pour
- accepter des niveaux de satisfaction ;
- augmenter la souplesse des modifications ;
- fournir plus d'informations en vue d'aider les décideurs.

°

° °

CHAPITRE V : L'ALGORITHME IMPLÉMENTÉ : GPSTEM

GPSTEM est le sigle de l'algorithme proposé par J. FICHEFET [8] et qui indique qu'il s'agit d'une fusion entre le goal programming et la méthode STEM. En premier lieu, nous allons resituer le contexte théorique de la démarche, puis nous détaillerons l'algorithme que nous avons implémenté. Les quelques différences avec la version de [8] seront ensuite expliquées et justifiées. Ce chapitre se terminera par le détail de l'analyse de sensibilité que nous avons inclu dans l'interface.

1. SOLUTIONS M-EFFICACES

La notion de solution efficace a été introduite au chapitre III.Elle a été retenue parce qu'elle n'impliquait aucune connaissance a priori sur les préférences relatives entre les objectifs. Les décideurs ont cependant une connaissance de l'entreprise. C'est ce qui a déjà justifié l'importance des cibles ou niveaux de satisfaction qu'ils sont à même de fournir. La notion de solution efficace va donc être affinée pour prendre en compte ces niveaux.

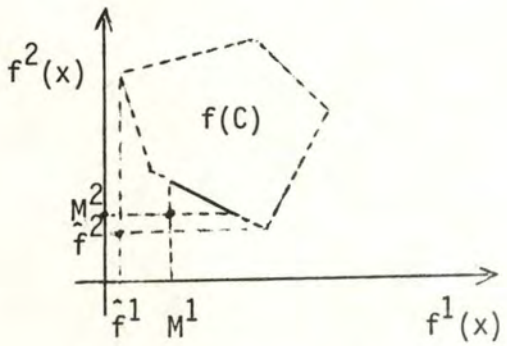
Soit $M \in \mathbb{R}^r$ le vecteur des niveaux de satisfaction

Définition : le point $x \in C$ est appelé une solution M-efficace ssi

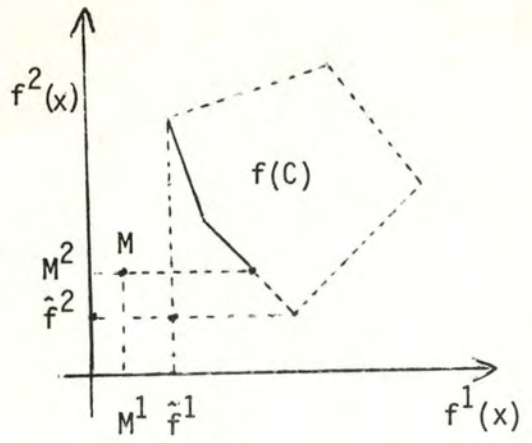
$$\nexists\, y \in C \text{ tel que } |f^i(y)-M^i| \leq |f^i(x)-M^i| \quad \forall i = 1, 2, \ldots, r$$

$$|f^i(y)-M^i| < |f^i(x)-M^i| \quad \text{pour au moins un } i$$

Interprétation géométrique (à deux dimensions)



le trait plein correspond à l'ensemble des solutions M-efficaces



le fait d'avoir pris $M^1 < \hat{f}^1$ n'augmente pas le nombre de solutions M-efficaces; ce résultat sera démontré dans le paragraphe suivant.

2. QUELQUES THEOREMES

Nous allons énoncer et démontrer quelques théorèmes pour justifier certaines propriétés des points M-efficaces et mieux comprendre les situations qui se présentent pendant le déroulement de l'algorithme.

2.1. Théorème 1

Soit $x \in C$

x est un point efficace ssi x est un point $\hat{f}$-efficace.

Démonstration

$$x \text{ est efficace} \Leftrightarrow \nexists y \in C \text{ tel que } \begin{cases} f^i(y) \leq f^i(x) & i = 1, 2, \ldots, r \\ f^i(y) < f^i(x) & \text{pour au moins un } i \end{cases}$$

$$\Leftrightarrow \nexists y \in C \text{ tel que } \begin{cases} f^i(y) - \hat{f}^i \leq f^i(x) - \hat{f}^i & i = 1, 2, \ldots, r \\ f^i(y) - \hat{f}^i < f^i(x) - \hat{f}^i & \text{pour au moins un } i \end{cases}$$

$$\Leftrightarrow \nexists y \in C \text{ tel que } \begin{cases} |f^i(y) - \hat{f}^i| \leq |f^i(x) - \hat{f}^i| & i = 1, 2, \ldots, r \\ |f^i(y) - \hat{f}^i| < |f^i(x) - \hat{f}^i| & \text{pour au moins un } i \end{cases}$$

car $\hat{f}^i \leq f^i(x) \quad \forall\, x \in C$

$\Leftrightarrow$ x est $\hat{f}$-efficace.

2.2. Théorème 2

Une solution optimale de $\min_{x \in C} d_p(f(x),M)$ pour $1 \leqslant p < \infty$

est M-efficace

Démonstration

Soit x une solution optimale qui n'est pas M-efficace

$\exists y \in C$ tel que $|f^i(y)-M^i| \leqslant |f^i(x)-M^i|$ $\quad i = 1, 2, \ldots, r$

$|f^i(y)-M^i| < |f^i(x)-M^i|$ pour au moins un i

En élevant chaque inégalité à la puissance p et en sommant membre à membre, on obtient :

$$\sum_{i=1}^{r} |f^i(y)-M^i|^p < \sum_{i=1}^{r} |f^i(x)-M^i|^p$$

x n'est donc pas une solution optimale.

Remarque : le théorème reste encore valable si, en plus d'élever les inégalités à la puissance de p, on les multiplie par $\alpha^i > 0$ avant de faire la somme. C'est le cas où la distance possède des facteurs de pondération. Nous avons ainsi démontré que le Goal Programming fournissait des solutions M-efficaces.
Le résultat obtenu n'est plus valable, dans toute sa généralité, quand on prend la distance L_∞ , c'est ce qui nous fera opter plus tard pour la distance L_1 .

2.3. Théorème 3

Soit $M \in \mathbb{R}^r$, un vecteur cible proposé par les décideurs et tel que
$\exists k : M^k = \hat{f}^k$
Soit $N \in \mathbb{R}^r$, tel que $N^i = M^i - m.\delta^i_k$ $\quad i = 1, 2, \ldots, r$ et $m>0$

$\bar{x}$ est une solution optimale du problème (1) $\min_{x \in C} d_1(f(x) ,N)$

ssi

$\bar{x}$ est une solution optimale du problème (2) $\min_{x \in C} d_1(f(x) ,M)$

Démonstration

$\Rightarrow$ si $\bar{x}$ n'est pas une solution optimale de (2)

$\exists\, y \in C$ tel que $\sum_{i=1}^{r} |f^i(y)-M^i| < \sum_{i=1}^{r} |f^i(\bar{x})-M^i|$

$$\sum_{\substack{i=1\\ i\neq k}}^{r} |f^i(y)-N^i| + |f^k(y)-M^k| < \sum_{\substack{i=1\\ i\neq k}}^{r} |f^i(\bar{x})-N^i| + |f^k(\bar{x})-M^k| \quad (a)$$

or, $M^k = \hat{f}^k \leq f^k(x) \quad \forall\, x \in C$

donc $|f^k(x)-M^k| = f^k(x)-M^k \quad \forall\, x \in C$

en ajoutant $m > 0$ à chaque membre de l'inégalité (a), on obtient :

$$\sum_{\substack{i=1\\ i\neq k}}^{r} |f^i(y)-N^i| + f^k(y)-M^k+m < \sum_{\substack{i=1\\ i\neq k}}^{r} |f^i(\bar{x})-N^i| + f^k(\bar{x})-M_k+m$$

on a toujours $f^k(x)-M^k + m > 0 \quad \forall\, x \in C$

ce qui donne : $\sum_{i=1}^{r} |f^i(y)-N^i| < \sum_{i=1}^{r} |f^i(\bar{x})-N^i|$

et $\bar{x}$ n'est plus solution optimale de (1).

$\Leftarrow$ si $\bar{x}$ n'est pas solution optimale de (1)

$\exists\, y \in C$ tel que $\sum_{i=1}^{r} |f^i(y)-N^i| < \sum_{i=1}^{r} |f^i(\bar{x})-N^i|$

$$\sum_{\substack{i=1\\ i\neq k}}^{r} |f^i(y)-M^i| + |f^k(y)-N^k| < \sum_{\substack{i=1\\ i\neq k}}^{r} |f^i(\bar{x})-M^i| + |f^k(\bar{x})-N^k|$$

or $N^k < f^k(x) \quad \forall\, x \in C$

donc $|f^k(x)-N^k| = f^k(x)-N^k$

$$= f^k(x)-M^k+m \quad \forall\, x \in C$$

de même $f^k(x)-M^k \geq 0 \quad \forall\, x \in C$

on obtient : $|f^k(x)-N^k| = |f^k(x)-M^k|+m \quad \forall\, x \in C$

$\bar{x}$ ne peut donc pas être solution optimale de (2).

Les théorèmes 2 et 3 justifient la deuxième remarque de l'interprétation géométrique. Non seulement, le fait de donner une cible plus petite que l'optimum d'une fonction n'amène pas de solutions efficaces supplémentaires, mais le compromis obtenu est également le même dans les deux cas.

2.4. Théorème 4

Si $M \notin f(C)$ et si $x \in C$ est M-efficace,
alors $f(x)$ appartient à la frontière de $f(C)$.

Démonstration

On a déjà vu que $f(C)$ est convexe et compact :
si $f(x)$ n'appartient pas à la frontière de $f(C)$, il appartient à Int $(f(C))$, l'intérieur de $f(C)$.

Or Int $(f(C))$ est ouvert, donc il existe une boule ouverte de centre $f(x)$ et de rayon 2ε, $B(f(x), 2\varepsilon)$, telle que
$\forall\, g \in B(f(x), 2\varepsilon) : g \in f(C)$

M ne peut se trouver dans la boule B, par hypothèse.
En conséquence, $\exists k$ tel que $f^k(x) - M^k > 2\varepsilon$

Considérons : $g \in f(C)$ et tel que :
$g^i = f^i(x) \quad i = 1, 2, \ldots, r$ et $i \neq k$
$g^k = f^k(x) - \varepsilon$
on a bien $g \in B(f(x), 2\varepsilon)$
de plus : $|g^k - M^k| = |f^k - \varepsilon - M^k| = f^k - M^k - \varepsilon$ car $f^k - M^k > \varepsilon$
$|f^k - M^k| - \varepsilon$ car $f^k - M^k > 0$
et x n'est plus une solution M-efficace.

2.5. Théorème 5

Si $M \in f(C)$, alors $\forall\, x \in C$ tel que $f(x) = M$, x est M-efficace.

Démonstration

Comme $M \in f(C)$, il existe au moins un point $x \in C$
tel que $f(x) = M$
si x n'est pas M-efficace
$\exists y \in X$ tel que
$|f^i(y) - M^i| \leq |f^i(x) - M^i| = 0 \quad i = 1, 2, \ldots, r$
$|f^i(y) - M^i| < |f^i(x) - M^i| = 0$ pour au moins un i
La dernière relation est impossible.

2.6. Théorème 6

Si y est une solution optimale de

$$\min_{x \in C} d_1 (f(x),M) = \min_{x \in C} \sum_{i=1}^{r} |f^i(x)-M^i|$$

alors y est aussi solution optimale de

$$\min_{x \in C} \sum_{\substack{i=1 \\ i \neq j}}^{r} |f^i(x)-M^i| + |f^j(x)-M^j-\Delta M^j|$$

avec $0 \leqslant \Delta M^j \leqslant f^j(y)-M^j$

Démonstration

Si y n'est pas solution optimale du deuxième problème,

$\exists z \in C$ tel que

$$\sum_{\substack{i=1 \\ i \neq j}}^{r} |f^i(z)-M^i| + |f^j(z)-M^j-\Delta M^j| < \sum_{\substack{i=1 \\ i \neq j}}^{r} |f^i(y)-M^i| + |f^j(y)-M^j-\Delta M^j|$$

Or $- |f^j(z)-M^j-\Delta M^j| \geqslant |f^j(z)-M^j| - |\Delta M^j|$

$- |f^j(y)-M^j-\Delta M^j| = f^i(y)-M^j-\Delta M^j$ car $\Delta M^j \leqslant f^j(y)-M^j$

$= |f^j(y)-M^j| - |\Delta M^j|$ car $0 \leqslant \Delta M^j$ et $0 \leqslant f^j(y)-M^j$

On obtient :

$$\sum_{i=1}^{r} |f^i(z)-M^i| - |\Delta M^j| < \sum_{i=1}^{r} |f^i(y)-M^i| - |\Delta M^j|$$

$$\sum_{i=1}^{r} |f^i(z)-M^i| < \sum_{i=1}^{r} |f^i(y)-M^i|$$

et y n'est plus solution optimale du premier problème.

Ce théorème aura une grande influence sur le choix des modifications à introduire, au cours du déroulement de l'algorithme. En effet, il prouve qu'une modification (ΔM^j) n'influence pas le problème tant qu'elle ne dépasse pas la valeur $f^j(y)-M^j$, c'est-à-dire la différence entre la valeur actuelle du compromis pour cet objectif ($f^j(y)$) et la valeur de son niveau de satisfaction.

3. L'ALGORITHME

3.1. Présentation

#1 Pour chaque i = 1, 2, ...,r successivement, calculer $\hat{x}_i$ solution optimale de $\min_{x \in C} f^i(x)$

à chaque itération, constuire la colonne P_i de la matrice P

$P_i^j = f^j(\hat{x}_i)$ j = 1, 2, ..., r

#2 Calculer $\alpha^i = \left| \frac{P^i_i}{P^i_i - m^i} \right| . \sqrt{\sum_{j=1}^{n} (C^i_j)^2}$ $\quad i = 1, 2, \ldots, r$

avec m^i = la plus grande valeur dans la ligne i de P

#3 Demander les niveaux de satisfaction $M^i \quad i = 1, 2, \ldots, r$

#4 k := 0; calculer x_k solution optimale de

$$\min z = \sum_{i=1}^{r} (y^{+i} + y^{-i})$$

$$\text{s.c } f^i(x) + \alpha^i y^{+i} - \alpha^i y^{-i} = M^i$$

$$x \in C,\ y^{+i} \geqslant 0,\ y^{-i} \geqslant 0 \quad i = 1, 2, \ldots, r$$

#5 $y^{+i} = 0 \quad \forall\ i = 1, 2, \ldots, r$

| oui : passer à l'étape suivante
| non : fournir x_k; STOP

#6 donner les valeurs prises par les fonctions économiques

| elles sont satisfaisantes : fournir x_k; STOP
| sinon: passer à l'étape suivante

#7 $y^{-i} = 0 \quad \forall\ i = 1, 2, \ldots, r$

| oui : passer à l'étape suivante
| non ; le décideur l désire une modification R^l de son objectif :

$$M^l := M^l + R^l;\ k := k+1;$$

soit x_k la solution après cette modification du second membre ;
aller en #5

#8 prendre un vecteur de perturbation des niveaux de satisfaction :

ΔM tel que $\Delta M^i \geqslant 0$, i = 1, 2, ..., r

Résoudre le problème paramétrique

$$\min z = \sum_{i=1}^{r} (y^{+i} + y^{-i})$$

$$\text{s.c. } f^i(x) + \alpha^i y^{+i} - \alpha^i y^{-i} = M^i - \Theta . \Delta M^i$$

$$x \in C,\ y^{+i} \geqslant 0,\ y^{-i} \geqslant 0 \quad i = 1, 2, \ldots, r$$

qui garde z = 0

| si $\Theta = 0$, il n'y a pas d'amélioration; retourner en #7 , deuxième partie
| sinon $M^i := M^i - \Theta \Delta M^i \quad i = 1, 2, \ldots, r$

soit x_k la solution après la paramétrisation; aller en #6

3.2. Commentaires

3.2.1. Les poids

A l'étape 2, on propose de calculer des facteurs de normalisation α^i. L'idée avait été introduite dans STEM [4].

Supposons que dans un problème, il existe les objectifs suivants :
- le chiffre d'affaire de l'entreprise, exprimé en francs;
- le temps d'inoccupation des machines, exprimé en pourcentage du temps total.

La première fonction aura des valeurs comprises entre zéro et plusieurs millions, tandis que la seconde sera limitée par zéro et cent. Une variation de dix unités n'a pas la même signification dans les deux cas.

D'autre part, si on a deux fonctions d'une même ordre de grandeur, et que la première est a peu près constante, ses variations prennent moins d'importance.

Les coefficients proposés permettent de tenir compte des remarques précédentes :

terme 1 $\left| \dfrac{p_i^i}{p_i^i - m^i} \right|$

m_i est la plus grande valeur dans la ligne i de la matrice P, c'est-à-dire la plus grande valeur prise par $f^i(x)$ pour les points correspondants aux solutions optimales des r fonctions.
L'importance des variables d'écart y^{+i} et y^{-i} augmente avec l'importance des écarts. On prend une variation relative pour tenir compte de la première remarque.

terme 2 $\sqrt{\sum_{j=1}^{n} (C_j^i)^2}$

ce coefficient sert à la normalisation des variables d'écart par rapport à la fonction économique.
C'est aussi dans le but de tenir compte de la première remarque.

De plus, ces poids qui viennent d'être introduits vérifient la propriété suivante : invariance des variables d'écart pour un changement d'échelle positif de la fonction économique.

Démonstration

Soit $f^i(x) + \alpha^i y^{+i} - \alpha^i y^{-i} = M^i$

si $f^i(x)$ devient $f'^i(x) = a.f^i(x) = \sum_{j=1}^{n} (a.c^i_j).x^j$; $a>0$

alors $\alpha'^i = a \cdot \alpha^i$

$M'^i = a \cdot M^i$ car $f^i(x)$ est linéaire et $a>0$

et on a encore $f'^i(x) + \alpha'^i y^{+i} - \alpha'^i y^{-i} = M'^i$

ce sont les mêmes variables d'écart.

Dans STEM, le poids α^i étaient ramenés entre 0 et 1. Ce qui, pour les auteurs, pouvait faciliter les comparaisons entre différentes stratégies de poids. Cependant, ils perdent de cette manière, la dernière propriété citée ci-dessus.

Ce schéma de normalisation des variables d'écart a été adopté dans l'implémentation, mais, dans le but de laisser aux utilisateurs la possibilité de choisir, deux autres techniques supplémentaires ont été retenues : les poids entre 0 et 1 et les coefficients tous égaux à 1, ce qui rend à nouveau les objectifs indiscernables.

3.2.2. La recherche du compromis

Lors du calcul d'un compromis, le problème général est :

$$\min_{x \in X} \; d_p(f(x),M) \text{ avec } 1 \leqslant p < \infty \qquad \text{(voir théorème 2)}$$

Dans le cas qui nous préoccupe, p aura la valeur 1

De plus, une distance peut être pondérée

$$d_p(\lambda,f(x),M) = \left(\sum_{i=1}^{r} (\lambda_i)^p . \; | f^i(x)-M^i |^p \right)^{1/p}$$

Ici, le vecteur de poids est égal à $(1/\alpha^1, 1/\alpha^2, \ldots, 1/\alpha^r)$, et on obtient : $\min_{x \in X} \sum_{i=1}^{r} \frac{1}{\alpha^i} |f^i(x)-M^i)|$

La généralisation du théorème 2 est immédiate pour autant que $1/\alpha^i > 0$, ce qui est bien le cas

Les équations complètes sont donc : $\min \sum_{i=1}^{r} (y^{+i} + y^{-i})$

$$\text{s.c } f^i(x) + \alpha^i y^{+i} - \alpha^i y^{-i} = M^i$$

$$x \in C, \; y^i \geqslant 0, \; y^{-i} \geqslant 0$$

Interprétation graphique des variables d'écart :

3.2.3. Etape 5

S'il existe un $y^{+i} > 0$, la solution obtenue par ce décideur est meilleure que ce qu'il espérait obtenir.

Comme les décideurs ont déjà une connaissance de la firme et qu'ils pourront en plus consulter la matrice de paiement avant de fournir les cibles, le traitement de ce cas n'est pas pris en charge par l'algorithme.

3.2.4. Etape de modification des niveaux de satisfaction

A l'étape précédente de l'algorithme, les décideurs ont pu évaluer le compromis obtenu. Si certains d'entre-eux ne sont pas satisfaits, on voudrait leur permettre d'élaborer une solution voisine de la première.

Deux cas peuvent se présenter : les objectifs prennent tous des valeurs égales aux cibles ($y^{-i} = 0$,i = 1, 2, ...,r) ou bien seulement quelques objectifs ont atteint leur niveau de satisfaction. La première situation est traitée à l'étape suivante.

Un moyen d'obtenir un autre compromis est de demander à un décideur l qui est déjà satisfait de relâcher sa cible d'une quantité R^l, dans le but de permettre une amélioration pour d'autres décideurs. Le théorème 6 a montré que pour entraîner un changement de solution, cette modification devrait dépasser $f^l(x) - M^l$, ce qui est en particulier vrai si ces deux valeurs sont confondues.

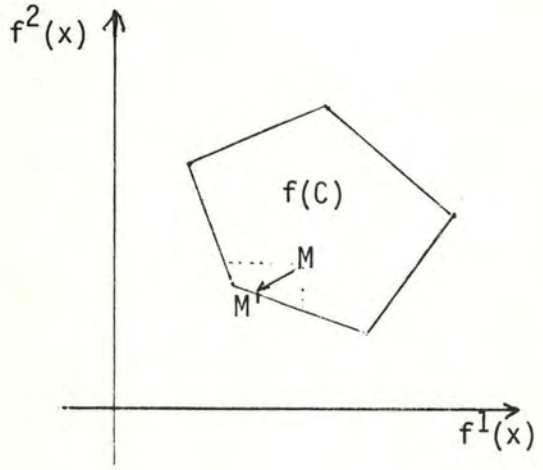
Il faut remarquer que rien n'impose dans l'algorithme que la modification à apporter soit l'augmentation d'un des niveaux de satisfaction, mais que c'est plutôt l'idée intuitive que l'on a d'un compromis entre deux objectifs conflictuels : pour que l'un des deux s'améliore, il faut que l'autre subisse une dégradation. C'est pourquoi, d'autres modifications seront aussi acceptées : diminuer une cible pour corriger une modification introduite auparavant, par exemple. De même, la restriction d'une seule modification par objectif est supprimée.

Le but de cette étape était d'explorer le voisinage du compromis actuel. Nous avons déjà éliminé les contraintes sur le genre et le nombre des modifications. Le point supplémentaire est de fournir une aide pour le choix de la modification. Ce point va être développé au paragraphe 6.

3.2.5. Etape d'amélioration globale

La cible est confondue avec le compromis et les utilisateurs désirent voir s'il y a moyen de "faire mieux". C'est principalement pour diminuer le nombre de passages à l'étape 7 qu'il a été prévu un traitement automatique de ce cas.

figure



L'idée est la suivante : fournir un nouveau point M' qui soit meilleur pour chaque décideur. Ce qui peut se formuler par : chercher $\hat{\theta}$ maximum tel que $M - \theta \Delta M \in f(C)$; $M' = M - \hat{\theta} \Delta M$. Comme M' appartient à f(C), il existe aussi un point x' de C dont l'image par f est M'. Il suffit de retourner à l'étape de présentation du compromis avec cette nouvelle solution. S'il n'est pas possible de trouver $\theta > 0$, il faut demander aux décideurs de préciser eux-mêmes les modifications qu'ils voudraient apporter au compromis actuel.
Pour cela, retourner à l'étape 7.

Le choix du vecteur ΔM sera exposé plus loin. On peut cependant déjà noter que son importance n'est que locale puisque, grâce à l'étape d'étude du voisinage du compromis, il est possible d'explorer toutes les solutions efficaces du problème.

4. MODIFICATIONS APPORTEES A LA PREMIERE VERSION

4.1. Solution efficiente d'équilibre

En cas d'impossibilité pour les décideurs de construire un compromis acceptable, J. FICHEFET [8] proposait une solution efficiente d'équilibre, calculée comme suit :

- après le calcul du compromis, chercher $x^{\star}_j$, J = 1, 2, ...,r tel que

  z soit nul dans le problème paramétrique

$$\min z = \sum_{i=1}^{r} (y^{+i} + y^{-i})$$

$$\text{s.c } f^i(x) + \alpha^i y^{+i} - \alpha^i y^{-i} = M + \Theta \delta^i_j \qquad i = 1, 2, \ldots, r$$

$$x \in C,\ y^{+i} \geq 0,\ y^{-i} \geq 0$$

- former B telle que $B^i_j = f^i(x^{\star}_j)$ $\qquad i,j = 1, 2, \ldots,r$

- déterminer un équilibre du jeu bimatriciel (P,B), c'est-à-dire un couple $(\bar{u}, \bar{v})$ tel que

$$u' \; P \; \bar{v} \geq (\bar{u})' \; P \; \bar{v}$$

$$(\bar{u})' \; B \; v \geq (\bar{u})' \; B \; \bar{v} \qquad \forall\, u, v \in \mathbb{R}^r$$

- résoudre $\min_{x \in C} \sum_{i=1}^{r} \bar{u}^i \, . \, f^i(x)$

  dont la solution x_e est la solution efficiente d'équilibre.

Cette solution peut servir de stimulus aux décideurs pour poursuivre la recherche d'un compromis. C'est là son principal avantage. Nous ne l'avons cependant pas adoptée pour deux raisons :

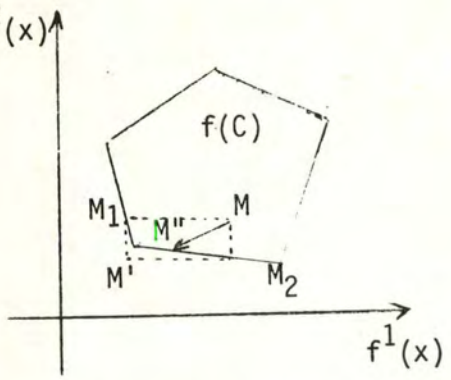
1) Le calcul de cette solution est fait avant chaque étape de modification; cela entraîne une lourdeur excessive de l'interface qui nuit au caractère opérationnel qui a été fixé comme objectif.

2) La souplesse de l'étape de modification des cibles nous paraît suffisante pour donner aux décideurs l'occasion de préciser leurs désirs, on plusieurs étapes, avec retour en arrière, ...

4.2. Amélioration des cibles

L'idée de cette étape était présente dans [8], ainsi qu'une possibilité de résolution.

figure



Pour chaque i = 1, 2, ..., r résoudre le problème paramètrique

$$\min \; z = \sum_{j=1}^{r} (y^{+j} + y^{-j})$$

$$\text{s.c. } f^j(x) + \alpha^j y^{+j} - \alpha^j y^{-j} = M^j - \Theta^i \delta_i^j$$

$$x \in C,\ y^{+j} \geqslant 0,\ y^{-j} \geqslant 0 \qquad j = 1, 2, \ldots, r$$

qui garde z nul.

Si $\hat{\Theta}^i$ reste nul pour chaque i, il n'y a pas d'amélioration possible. Sinon, on remplace M^i par $M^i - \hat{\Theta}^i$ et on retourne à l'étape de calcul du compromis (avec M' comme cible, dans la figure).

Cela revient à rechercher pour chaque objectif, la diminution maximum qui garde M dans f(C), (on trouve successivement M_1 et M_2). Cette façon de procéder permet de donner des informations supplémentaires à chaque décideur pour des compromissions futures éventuelles.

Dans la version proposée ici, nous avons voulu diminuer les temps de calcul en évitant les r paramétrisations successives et le retour au calcul du compromis. Pour cela, il falait trouver un point M" qui soit déjà un compromis et que l'on soumettra directement aux décideurs. Partant de M, on ne fera plus

une paramétrisation dans chaque direction, mais une paramétrisation globale qui fournira le point M" :
chercher Θ maximum tel que

$$\min z = \sum_{i=1}^{r} (y^{+i} + y^{-i})$$

$$\text{s.c. } f^i(x) + \alpha^i y^{+i} - \alpha^i y^{-i} = M^i - \Theta \Delta M^i$$

$$x \in C \ , \ y^{+i} \geqslant 0, \ y^{-i} \geqslant 0, \quad i = 1, 2, \ldots, r$$

et qui garde z nul.

5. TABLEAU SIMPLEXE INITIAL ET MIS A JOUR

Nous allons récrire le problème en nous référant à Orchard-Hays [13]. Toutes les lignes d'un programme linéaire s'écrivent sous forme d'équation :

$$\mu^i + \sum_{j=1}^{n} d_j^i x^j = b^i$$

où
- μ^i est la variable logique, ou variable d'écart pour cette ligne.
- x^j sont les variables structurelles du problème.
- d_j^i sont les coefficients de la matrice (données du problème).
- b^i est le second membre de cette ligne, souvent appelé RHS (right hand side).

Toutes les variables possèdent un type. En particulier, pour les variables logiques, le type fixe le sens de l'inégalité tel qu'il a été vu auparavant.

FREE : la variable logique peut prendre toutes les valeurs possibles, il s'agit d'une fonction économique.

PLUS : cela correspond à une contrainte $\sum_{j=1}^{n} d_j^i x^j \leqslant b^i$

MINUS : $\sum_{j=1}^{n} d_j^i x^j \geqslant b^i$

ZERO : $\sum_{j=1}^{n} d_j^i x^j = b^i$

RANGE = a : $b^i - a \leqslant \sum_{j=1}^{n} d_j^i x^j \leqslant b^i$

Un problème est complètement déterminé quand on donne toutes ses équations et le type de toutes ses variables.

5.1. Tableau initial

Toutes les équations peuvent se représenter sous forme d'un tableau simplexe. Les variables logiques forment la base de départ.

μ^0	$\mu^1 \ldots \mu^r$	$\mu^{r+1} \ldots \mu^{r+m}$	$\mu^{r+m+1} \ldots \mu^{2r+m}$	$x^1 \ldots x^n$	$y^{+1} \ldots y^{+r}$	$y^{-1} \ldots y^{-r}$	RHS
1	0 ...			0	1 ... 1	1 ... 1	0
0	1		0	$c_1^1 \ldots c_n^1$	0 ...		0
0	... 1 0 ...		0	$c_1^r \ldots c_n^r$	0 ...		0
0	0 1		0	$a_1^1 \quad a_n^1$	0 ...		0
0	...	1 0 ...	0	$a_1^m \quad a_n^m$	0 ...		0
0	...	0 1	0	$c_1^1 \quad c_n^1$	α^1	0 $-\alpha^1$	0
0	...		1	$c_1^r \quad c_n^r$	0	α^r 0 $-\alpha^r$	M^1 ⋮ M^r

5.2. Tableau mis à jour

A une itération particulière, il existe toujours $2r + m + 1$ variables dans la base courante. Les variables $\mu^0, \mu^1, \ldots, \mu^r$ y sont toujours puisqu'elles ont le type (FREE). Le tableau, mis à jour en terme de la base actuelle est alors

μ^0	$\mu^1 \ldots \mu^r$	$\mu^{r+1} \ldots \mu^{r+m}$	$\mu^{r+m+1} \ldots \mu^{2r+m}$	$x^1 \ldots x^n$	$y^{+1} \ldots y^{+r}$	$y^{-1} \ldots y^{-r}$	RHS
1	0 ... 0	q_1^0	q_{m+1}^0	q_{m+r+1}^0	$q_{m+n+r+1}^0$	q_{m+n+3r}^0	-z
0	1						$-f^1(x)$ ⋮ $-f^r(x)$
	1						B^1 ⋮ B^{m+r}
		q_1^{m+2r}				q_{m+n+3r}^{m+2r}	

6. ANALYSE DE SENSIBILITE

Lors de l'étape de modification des niveaux de satisfaction, comment choisir le décideur l et la quantité R^l ? Il faudrait une certaine estimation des taux de substitution entre les différents objectifs, pouvoir connaître

les bornes de variation pour un accroissement unitaire de l'objectif l, par exemple.

6.1. Bornes de variation

L'idée de cette solution était proposée dans STEM [2]. Supposons que les r fonctions économiques sont présentes dans le tableau simplexe lors de la recherche du compromis. Elles existaient auparavant pour la construction de la matrice de paiement. Il suffit de les garder. On obtient un tableau semblable à (5.2). Considérons les r lignes correspondant aux r objectifs.

Pour que l'objectif l augmente, il faut que ce soit un vecteur hors base ayant une coefficient $q_s^l > 0$ qui rentre dans la nouvelle base. Mais si une variable entre avec un niveau $1/q_s^l$ (pour avoir une augmentation unitaire de $f^l(x)$), elle entraîne des modifications dans les autres fonctions économiques, dépendant du signe des coefficients q_s^i, $i = 1, 2, \ldots, r$, $i \neq l$

si $q_s^i > 0$, la fonction i va croître de $\frac{q_s^i}{q_s^l}$

si $q_s^i < 0$, le fonction i va décroître de $|\frac{q_s^i}{q_s^l}|$, c'est-à-dire

varier de $\frac{q_s^i}{q_s^l}$

donc :

- la borne supérieure de variation de l'objectif i, pour une variation unitaire de l'objectif l, soit $\cup_l^i$, est égale à :

$$\max_{s \in S_T} \frac{q_s^i}{q_s}$$

où S_T est l'ensemble des indices des variables hors base dont $q_s^l > 0$.

- la borne inférieure de variation de l'objectif i, pour une variation unitaire de l'objectif l, soit L_l^i, est égale à

$$\min_{s \in S_T} \frac{q_s^i}{q_s^l}$$

interprétation : si $\cup_l^i < 0$ l'objectif i va décroître si l augmente, et on a $L_l^i \leq$ variation $\leq \cup_l^i < 0$.

si $L_l^i > 0$ l'objectif i va croître en même temps que l, et on a $0 < L_l^i \leq$ variation $\leq \cup_l^i$

sinon ($L_l^i \leq 0 \leq \cup_l^i$) la variation de l'objectif i, dont on connaît cependant les bornes, va dépendre du vecteur entrant dans la base.

On pourrait faire de même si l'objectif l doit subir une diminution ; dans ce cas, il suffit de considérer S_l, l'ensemble des indices des variables hors base dont $q_s^l < 0$.

6.2. Approximation linéaire de la solution

Le paragraphe précédent a cherché à établir des bornes de variation des objectifs, portant d'une variation unitaire de l'un d'eux. Mais on a vu qu'il existait des cas où le sens de variation d'un objectif i n'est pas univoque pour une modification unitaire de l'objectif l.

On peut penser à chercher une autre évaluation de ces taux de substitution. De plus, l'estimation précédente portait sur les objectifs eux-mêmes alors que la modification effective porte sur les seuils de satisfaction.

Avoir utilisé le terme "taux de substitution" peut suggérer une autre solution. En général, il résulte de l'algorithme du simplexe que si on modifie le second membre d'une contrainte : $b^i \to b^i + \lambda^i$, alors le second membre, exprimé en terme de la base actuelle est donné par $\bar{b}^j + \lambda^i \bar{q}_i^j$ où $\bar{q}_i^j$ est l'élément j de la colonne correspondant à la variable logique i.

Ici, si on admet une modification de la cible $M^l \to M^l + R^l$, les fonctions économiques prennent les valeurs $f^i - R^l . q_{m+l}^i$ $i = 1, 2, \ldots, r$.

Ce deuxième membre est encore réalisable si chaque composante $B^i + R^l q_{m+l}^{r+i}$ vérifie le type de la variable en base correspondante. Par exemple, pour des variables de type (PLUS), c'est-à-dire positives ou nulles, il faut que $B^i + R^l q_{m+l}^{r+i}$ reste positif. La solution est donc encore optimale puisque les coefficients de la fonction économique n'ont pas été modifiés Sinon, on peut fournir aux utilisateurs les bornes de validité pour les modifications R^l. Pour le cas particulier considéré, il faut

$$\max_{q^{r+i}_{m+1} > 0} \left(\frac{B^i}{q^{r+i}_{m+1}}\right) \leq R^l \leq \min_{q^{r+i}_{m+1} < 0} \left(-\frac{B^i}{q^{r+i}_{m+1}}\right)$$

Les autres types se traitent de la même façon (voir Orchard-Hays [13] p. 143). Ces seuils sont ceux que l'on obtient après une analyse de sensibilité du second membre.

D'un point de vue pratique, tant que les utilisateurs respectent ces bornes, on peut leur donner les valeurs réelles que prendront les objectifs, en connaissant les valeurs actuelles et la colonne, mise à jour, q_{m+1}.

En dehors de ces bornes, les valeurs fournies sont des approximations linéaires des valeurs que prendront les objectifs après le ou les changements de base entraînés par la modification.

C'est la simplicité et l'élégance de cette méthode, ainsi que la qualité des informations données à l'utilisateur qui nous l'a fait retenir pour l'implémentation.

7. LE VECTEUR DE PERTURBATION

La solution la plus simple est de prendre pour ΔM le vecteur $(-1, -1, \ldots, -1)' \in \mathbb{R}^r$. Tous les objectifs vont diminuer de la même quantité. On a déjà souligné en 3.2.1. les ennuis que peut avoir cette approche. De plus, une pondération statique (c'est-à-dire qui ne dépend que des valeurs fixes : coefficients des fonctions économiques, solutions optimales, ..., et pas de la solution actuelle) ne suffit pas non plus.

Il serait intéressant d'avoir une approximation de la quantité dont peut diminuer M^i tout en gardant le point M dans f(C), et cela sans faire les paramétrisations. On sait que si M atteint la frontière de f(C), pour la dépasser, la variable y^{-i} deviendra strictement positive, ce qui correspond au moins à un changement de base dans le tableau simplexe. La quantité que l'on cherche à évaluer est donc plus grande ou égale à la borne inférieure

$$\max_{q^{r+i}_{m+1} > 0} \left(\frac{B^i}{q^{r+i}_{m+1}}\right)$$

introduite au paragraphe précédent. Dans le cas présent, on prendra chaque composante de $-\Delta M$ égale à sa borne respective.

Comme l'influence de ce choix n'est que locale, il n'est pas possible d'opter définitivement pour l'une ou l'autre solution. C'est pourquoi, le type de vecteur ΔM pourra être choisi par les utilisateurs.

° ° °

DEUXIEME PARTIE :

LE SOFTWARE DE PROGRAMMATION LINEAIRE

INTRODUCTION

Le but de ce mémoire était de créer un interface de programmation multicritère à intégrer dans un software de programmation linéaire. Après avoir élaboré l'algorithme dans la partie précédente, nous allons étudier l'outil qui sera utilisé, et mettre en évidence les possibilités offertes dans la résolution d'un tel problème.

L'algorithme proposé fait partie d'une classe d'extensions de la programmation linéaire qui utilise l'optimisation de sous-problèmes. L'aspect interactif veut fournir des indications sur l'état actuel et sur l'évolution future, accepter des modifications et refaire une optimisation, soit en utilisant l'algorithme primal, soit en effectuant des paramétrisations. Dans ce dernier cas, il importe de pouvoir considérer la solution obtenue comme la solution courante, sans retenir la manière qui l'a amenée.

Dans ces étapes, l'accent est mis principalement sur l'efficacité des techniques retenues pour donner pleinement son sens au caractère interactif de la résolution.

D'autres actions propres à cet algorithme seront aussi reprises dans la présentation du programme produit :

- calculer l' optimum de plusieurs fonctions économiques;
- accéder aux coefficients de la matrice : valeurs de départ et valeurs mises à jour;
- ajouter des lignes et des colonnes à la matrice donnée par l'utilisateur;
- résoudre des problèmes paramétriques et postoptimaux;
- modifier des éléments.

Des actions supplémentaires apparaîtront ultérieurement lors de l'étude des spécifications du langage, elles concernent principalement les points de reprises.

Dans le premier chapitre, nous présentons le programme-produit MPS/66 du constructeur Honeywell Bull. Le second chapitre donnera les solutions qui ont été retenues pour les différents problèmes qui viennent d'être cités.

CHAPITRE I : PRÉSENTATION DU MPS/66

Un software de programmation linéaire peut être étudié par rapport à plusieurs points de vue, notamment :

- le langage de contrôle;
- la gestion des entrées et des sorties;
- la richesse des algorithmes;
- la taille du problème et la vitesse des calculs;
- la maintenance du programme.

De par le but que nous poursuivons, les trois premiers points seront principalement développés . Pour situer les autres caractéristiques, on peut dire que la taille maximum d'un problème standard est de 4095 lignes et de 262.000 colonnes; cette taille maximum peut encore être augmentée grâce à des techniques particulières telles que la décomposition de Dantzig-Wolfe qui admet un maximum de 12 blocs avec au plus 4.095 lignes par bloc.
Mais rien dans l'algorithme n'est directement influencé par cette taille maximum. D'autre part, l'interface à réaliser sera intégré, quant à son déroulement, dans le programme-produit et il n'est nullement question de l'incorporer dans le code même du constructeur. C'est pourquoi, la maintenance du programme (découpe modulaire, accès aisé à la structure interne des algorithmes, ...) n'a pas été abordée.

Signalons enfin que la présentation développée ici est loin d'être exhaustive, mais qu'elle essaie plutôt de donner les concepts du MPS qui seront utilisés dans la suite du travail.

1. LE LANGAGE DE CONTROLE (Agenda Control Language)

Le but d'un langage de contrôle est d'amener les facilités disponibles dans les langages de programmation classiques tout en gardant l'optique résolution de programmes linéaires. En général, une commande est composée :
- d'une étiquette optionnelle;
- d'un verbe qui l'identifie;
- d'une suite de phrases éventuellement nulle et qui contient les paramètres de cette commande.

1.1. Appel de procédures

Il s'agit de l'appel aux différentes fonctions de création de matrices (CONVERT, SETUP, ...), de résolution du problème (PRIMAL, ...), d'impression de résultats (OUTPUT, ...).

1.2. Manipulation des contrôles de solution

1.2.1. Description

Il en existe de différents types. L'exemple qui sera retenu est influencé par les problèmes qui seront rencontrés.

- Contrôle de fréquence : introduisent des limites sur le nombre d'itérations de l'algorithme primal avant d'effectuer certaines actions.
  Exemple : FIV : fréquence d'inversion. La valeur standard est 100, ce qui signifie que, toutes les 100 itérations, il y a réinversion de la base courante.

- Tolérances : seuils pour les valeurs obtenues lors du contrôle des erreurs d'arrondi, du choix du pivot, d'entrée des données,...
  Exemple : TMX : borne sur le plus grand nombre en valeur absolue qui peut être accepté en entrée; sa valeur standard est 10^6.

- Demandes d'actions : effectuent un déroutement lorsque certaines conditions sont rencontrées pendant l'exécution de la procédure. On leur assigne le label d'une routine particulière (voir contrôle de séquence, 1.4.).
  Exemple : NOMAX : si dans une paramétrisation, le paramètre peut croître indéfiniment, on exécute la routine dont le label est contenu dans NOMAX.

- Arguments : comme le MPS accepte, par exemple, plusieurs fonctions économiques, plusieurs seconds membres dans une même matrice, il faut spécifier les vecteurs à retenir pour chaque optimisation.
  Exemples : OBJ : nom de la fonction économique;
  CRHS : nom du second nombre composé.

• Paramètres : règlent le fonctionnement des algorithmes.

Exemples : SCALE : facteur d'échelle pour la fonction économique courante;

THETA : coefficient multiplicatif du second membre composé.

• Switchs : permettent le choix de certains modes d'exécution; les valeurs booléennes qu'ils prennent sont notées ON et OFF.

Exemple : CRHSW : le second nombre courant est une combinaison linéaire de RHS et de THETA * CRHS

1.2.2. Changer et afficher les valeurs

Les opérations se font par des commandes communes à tous les contrôles de solution.

- positionner des valeurs particulières :

SET FIV = 50, NOMAX = LABEL1, OBJ = PROFIT

- remettre les valeurs standards :

RESET CRHSW

- affichage : impression sur le listing de toutes les valeurs :

STATUS

1.3. Manipulation des variables

Certaines variables de communication existent déjà dans le MPS; ce sont, par exemple, USER1 jusque USER9; de plus l'utilisateur peut en définir d'autres et les manipuler par des expressions du même genre que les expressions en FORTRAN, elles peuvent être booléennes ou arithmétiques suivant le type de la variable.

COMPUTE USER1 = USER1 + 5 * THETA

1.4. Contrôle de séquence

Pour dépasser l'exécution séquentielle des instructions, l'utilisateur a le choix entre plusieurs modes de contrôle :

- branchement inconditionnel :

GOTO < label >

où < label> est défini ailleurs dans le programme, par sa simple apparition dans le premier champ d'une instruction

ENDLP

clôture l'exécution du MPS.

- branchement conditionnel :

si une condition est réalisée, il y a branchement au label indiqué
LOGIC < expression booléenne >, < label >

- boucles :

permettent de programmer des itérations
TALLY < label >, < variable >, < maximum >, < incrément >

- sous-routines :

l'entrée dans la routine peut être le résultat d'une demande d'action (voir 1.2.1.), ou bien est due à l'exécution de
PERFORM < label >
le retour peut se faire suivant trois possibilités :

CURRENT : il y a retour à l'instruction qui a initialisé le transfert. C'est nécessaire pour les demandes d'inversion, par exemple, pendant l'exécution du primal.

NEXT : il y a retour à l'instruction suivante.

JUMP : il y a retour à l'instruction qui suit celle qui a initialisé l'exécution de la routine la plus externe, dans le cas de routines imbriquées.

1.5. Macro-langage

Il permet de regrouper certaines fonctions élémentaires, avec la possibilité d'actualisation de certains paramètres. On obtient ainsi des fonctions d'un niveau supérieur qui rencontrent mieux le problème à résoudre. Un des objectifs de l'implémentation est d'arriver à l'écriture d'un macro GPSTEM qui contiendra toute la logique et toutes les fonctions nécessaires à l'interface.

Passage des paramètres : il existe seulement des paramètres positionnels. Dans le prototype de la macro, ils s'écrivent #i, ou i est l'indice dans la liste des arguments réels.

1.6. Appel de procédures utilisateurs

Dans un premier temps, il faut constituer un programme exécutable avec les routines que l'on veut appeler à partir du MPS. Le lien entre ces routines et le reste du package est dynamique. Le programme utilisateur est chargé comme un overlay particulier de l'ensemble du programme-produit. Comme le lien est dynamique, cela ne nécessite pas de créer un programme commun

à l'aide d'un éditeur de lien. Cette solution amène une très grande souplesse de mise au point.
Par contre, le fait d'être chargé à chaque appel, peut influencer le style de programmation; il n'y a aucune zone permanente en mémoire entre les appels. Cet ennui se répercute sur les variables et les vecteurs de l'interface, mais aussi sur ses zones de description de fichiers.

Les arguments que peut recevoir le programme-utilisateur sont constitués des contrôles de solution et des variables de communication USERi. De plus, il peut avoir accès à certains fichiers externes qui sont lus ou écrits par le MPS.

C'est sur base des possibilités d'intervention entre le programme-utilisateur et le système que se juge la puissance de traitement que peut avoir l'insertion d'un tel programme.

2. LA GESTION DES ENTREES ET DES SORTIES

2.1. Notions sur les fichiers

Pour MPS, un fichier est une collection d'informations, identifié par un nom et situé sur un support physique connu du système d'exploitation. Ce support n'est déclaré qu'au moment de l'exécution et le lien se fait par un "file-code" XX, où X est un caractère alphanumérique. Ce support est donc également un fichier, niveau GCOS, qui peut contenir plusieurs fichiers, niveau MPS. Quand le contexte est suffisant pour lever toute ambiguïté, on emploiera simplement le terme "fichier".
La description complète d'un fichier est donnée par

< nom >/< file code >

Les fichiers-GCOS, manipulés par le MPS, peuvent se répartir en quatre classes :

- les fichiers d'entrée et de sorties accessibles à l'utilisateur (données, changements au problème, base, liste de vecteurs, ...) : IN, AI, ..., GI.
  La structure générale d'un fichier MPS sur un tel fichier est la suivante :

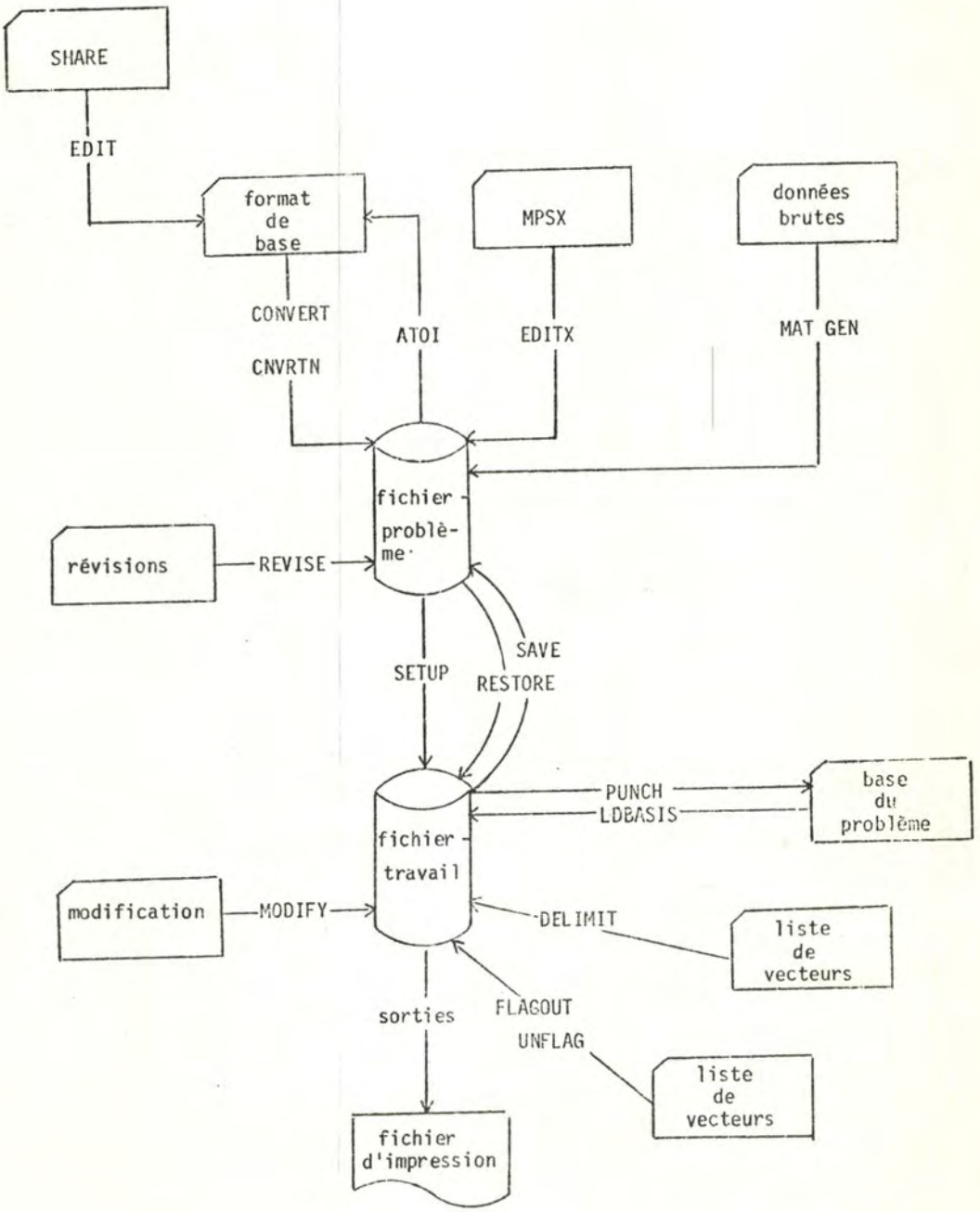
  FILE < nom >
  < données particulières >
  END ***

  Ils sont partageables avec les routines utilisateurs, chaque enregistrement est une image carte.

- les fichiers-problèmes : XP, PT ou VP (ils seront expliqués plus loin)
- les fichiers d'impression : SO et la sortie auxiliaire XO.
  Chaque enregistrement est l'image d'une ligne d'impression.
- les fichiers temporaires : pour les manipulations internes du système, ils ne sont jamais référencés par l'utilisateur.

2.2. Schéma d'évolution des données

Les verbes qui sont placés sur les axes du schéma sont les verbes qui commandent les différentes actions.



2.3. Les formats externes

2.3.1. Format MPS

Le but n'est pas de donner une liste complète du format de toutes les cartes de description de la matrice, mais plutôt de préciser les concepts et l'organisation générale de cette description.

On peut distinguer quatre parties principales :

- la déclaration des lignes de la matrice, avec leurs caractéristiques telles que :
  - le type : PLUS, MINUS, ZERO, FREE, RANGE, il donne le sens de la ou des inégalités
  - un facteur d'échelle, sauf pour les lignes FREE
    SCALE = < valeur >
    Ceci implique qu'on ne sait pas fixer le sens de l'optimisation d'une fonction économique lors de sa déclaration, mais que ce renseignement n'est donné qu'avant l'appel de l'algorithme, par le paramètre SCALE (voir 1.2.1.)
  - d'autres renseignements tels que les nombres d'éléments non nuls dans la ligne (pour des questions de vérification éventuelle).
- la déclaration des colonnes avec leurs caractéristiques.
  Les principes sont les mêmes que pour les lignes.
- la déclaration des éléments non nuls de la matrice.
  On introduit la valeur de chaque élément, qui est identifié par un nom de ligne et un nom de colonne.
- la déclaration du ou des vecteurs second(s) membre(s), appelé(s) RHS (right hand side). Le nom est donné par sa simple apparition et les éléments de la colonne sont identifiés par le nom de la ligne et du RHS correspondant.

Le format et l'ordre d'introduction des cartes sont libres et la procédure construisant la matrice en format interne est alors CONVERT. Il existe, à partir des mêmes éléments de description, un format fixe et un ordre déterminé, le tout donnant le format "ATOI", qui peut être traduit par CNVRTN, procédure beaucoup plus rapide car elle évite des tris et le balayage lexicographique des cartes.

Un fichier problème peut être réexprimé en format externe par le verbe ATOI, qui génère des cartes respectant ce mode.

2.3.2. Formats venant d'autres systèmes de programmation linéaire

- Format SHARE : il est converti en format externe MPS, puis la procédure classique peut être appliquée.
- Format MPSX : c'est le format utilisé dans le système de programmation linéaire d'IBM sur la série 360 et 370, ainsi que dans le MPS de la série 200 d'Honeywell.
  Ces données sont traduites directement en format interne.

2.3.3. Données brutes

Le système offre la possibilité de traiter des données brutes sans être forcé de les écrire soi-même dans le format externe. Le générateur de matrice construit, à partir de ces données, une matrice en format interne, en suivant les directives que l'utilisateur a exprimées dans un programme de génération.

2.4. Le fichier problème

2.4.1. Caractéristiques

Il peut contenir simultanément deux classes de fichiers MPS :

- des matrices en format interne : elles sont identifiées par le nom qui a été donné dans CONVERT (*).
- des solutions : reprenant l'état de la résolution atteint lors de leur création.

Un programme peut avoir deux fichiers-problèmes identifiés par les file-codes :

- XP : ce fichier ne peut être accédé qu'en lecture.
- PT ou VP : ce fichier peut être accédé en lecture et en écriture avec la distinction suivante :
  - PT : le fichier est vide au départ.
  - VP : le fichier contient déjà des informations à utiliser.

Dans les commandes du langage de contrôle, seuls XP et PT sont connus; la distinction supplémentaire n'est faite qu'au niveau des cartes contrôle GCOS.

(*) Les données d'un problème se trouvant dans l'état de matrice binaire sur le fichier-problème sont parfois dénommées, par abus de langage, "fichier-problème".

Les possibilités offertes par ces deux fichiers seront avantageusement exploitées dans le cas de solutions antérieures, de points de reprises, etc. Nous les examinerons dans la troisième partie quand nous introduirons le langage qui a été proposé pour l'interface.

Une dernière remarque pour dire que l'utilisateur n'est pas forcé de se définir un fichier-problème au niveau du système d'exploitation, le MPS se le réservant alors dynamiquement dans ses fichiers temporaires.

Vu la diversité des manières de créer une matrice en format interne, il est logique que l'interface ne prenne en charge les données qu'à partir de cette étape.

2.4.2. Les KJ

Pour l'utilisateur, les lignes et les colonnes de la matrice sont identifiées par un nom. Le MPS leur ajoute par après un numéro d'ordre, appelé KJ, qui lui permet une manipulation plus aisée de ces énormes matrices.

Les règles d'affectation de ces numéros d'ordre sont les suivantes :

- ils croissent séquentiellement à partir de 1.
- ils reprennent tous les vecteurs lignes, puis tous les vecteurs colonnes et finalement les RHS.
- à l'intérieur d'un groupe, les vecteurs sont classés suivant l'ordre de leur déclaration.

2.5. Le fichier de travail

Il peut exister plusieurs matrices en format binaire sur le fichier-problème, mais les algorithmes ne travaillent que sur une seule à la fois. La préparation de leur exécution est faite par le verbe SETUP qui constitue le fichier de travail. Ses principales fonctions sont :

- créer une base initiale composée de toutes les variables logiques à partir des statistiques, sur la matrice, qui ont été accumulées par les verbes tels que CONVERT;
- se réserver dynamiquement en mémoire centrale, des buffers, des régions de communication, ...

Lors de la constitution du fichier de travail, on peut également spécifier certaines options qui permettent d'améliorer la précision numérique au cours des itérations; par exemple :

AUTOSCALE : tous les coefficients de la matrice sont ramenés à un même ordre de grandeur pour éviter les écarts disproportionnés et favoriser la précision dans les inversions de matrices de base.

L'espace physique nécessaire au fichier de travail est réservé par le système à l'intérieur de ses fichiers temporaires; il n'y a donc pas de file-code.

2.6. Les changements au problème

Le système gère deux types de changements correspondant aux deux niveaux internes de la matrice du problème.

2.6.1. Révisions

Ce sont les changements qui sont apportés au fichier-problème Toutes les possibilités existent :

- insertion de lignes, de colonnes et de RHS
- suppression de lignes, de colonnes et de RHS
- remplacement des éléments de la matrice
- changement des caractéristiques des lignes et des colonnes.

En ce qui concerne la création de lignes et de colonnes, elles peuvent être déclarées comme combinaison linéaire d'autres vecteurs de la même classe. On peut demander l'insertion avant ou après les autres éléments de la même classe. Comme ces changements impliquent une réorganisation complète de la matrice, le système en crée une nouvelle.
En particulier, les KJ associés à certains vecteurs sont changés si il y a eu des insertions ou des suppressions avant eux.
En conséquence des règles de révisions, on constate que ces changements sont permanents.

2.6.2. Modifications

Elles affectent le fichier de travail et de ce fait, elles ont uniquement un caractère temporaire.

Les seuls changements permis sont :

- la modification d'éléments non nuls de la matrice;
- l'ajoute de nouveaux RHS en fin de série;

On peut immédiatement vérifier que ces changements ne modifient pas la structure de la matrice et les KJ restent identiques.

Ces modifications au fichier de travail le rendent incohérent par rapport au fichier duquel il est issu. Cette remarque prendra toute son importance dans le paragraphe suivant.

2.7. Les résultats intermédiaires

2.7.1. La solution complète

L'état complet de la solution courante peut être sauvé sur le fichier-problème par l'ordre SAVE. Il comprend les données de la solution, l'inverse de la base actuelle (sous forme des vecteurs ETA) et certaines régions de communication utiles.

Pour une utilisation ultérieure , il faut d'abord réinstaller le fichier de travail correspondant, puis charger la solution par RESTORE.

Les modifications, telles qu'elles ont été définies ci-dessus suppriment la possibilité d'utilisation d'une solution complète puisqu'il n'existe pas de fichier-problème correspondant au fichier de travail modifié.

2.7.2. La base

C'est une liste comprenant le nom de toutes les variables logiques et structurelles en base ainsi que de celles qui se trouvent hors-base mais à leur borne supérieure; c'est une modification apportée à la définion d'une base d'un tableau simplexe pour tenir compte des variables doublement bornées (voir Orchard-Hays [13] p. 28)

2.8. Les sous-ensembles de variables

2.8.1. Variables exclues de la solution

Certaines variables logiques et structurelles peuvent être laissées à un niveau nul, soit en les empêchant d'entrer dans la base, soit en leur donnant le type ZERO si elles s'y trouvent. La commande correspondante est FLAGOUT. La suppression de ce sous-ensemble se fait pas UNFLAG.

Il existe plusieurs manières de citer un sous-ensemble de variables :

- en les citant directement, avec la possibilité de donner simplement des masques sur les noms (noms qui se terminent par certains caractères, ...)
- en renseignant un fichier contenant une liste du type précédent. Cette possibilité peut être très intéressante puisqu'elle permet de spécifier dynamiquement, par le garnissage d'un fichier, un sous-ensemble de variables.

2.8.2. DELIMIT set

Ce sous-ensemble sert à restreindre les sorties de certains verbes qui demandent des impressions. Il est créé par le verbe DELIMIT.

Lors de l'exécution de plusieurs verbes DELIMIT, ce sous-ensemble ne peut être qu'étendu, et il n'est supprimé que par l'exécution de SETUP.

2.9. Les sorties

2.9.1. La solution

Demandée par le verbe OUTPUT, elle peut être imprimée ou gardée sur un fichier (IN, AI, ..., GI). Elle comprend les valeurs prises par les différentes variables logiques et structurelles, plus d'autres informations telles que le type de la variable, le coefficient correspondant de la fonction économique, ...

2.9.2. La matrice mise à jour

Il existe trois possibilités de sortir les éléments de la matrice :

- COLOUT : donne les éléments correspondant aux variables en base et aux variables structurelles hors base.

- INVOUT : donne les éléments correspondant aux variables en base et aux variables logiques hors base.
- ROWOUT : rassemble les deux types d'information, mais arrangés suivant les lignes de la matrice.

Toutes les sorties se font sur le fichier-imprimante SO. Pour tous les verbes, les lignes qui sont imprimées correspondent au DELIMIT set courant, tandis que les colonnes sont limitées à un sous-ensemble donné avec le verbe.

2.9.3. Les fichiers d'impresssion

Le MPS possède deux fichiers d'impression identifiés par

- SO : fichier standard
- XO : fichier auxiliaire, qui sert principalement quand on veut de gros volumes d'impressions, tels que par OUTPUT, et qui ne sont pas intercalés avec les impressions de contrôle du déroulement du programme.

3. RICHESSE DES ALGORITHMES

3.1. Algorithme primal du simplexe

Il calcule la solution optimale du problème dont la matrice a été mise en forme par SETUP. Ce calcul se décompose souvent en deux étapes :

- CRASH : pour améliorer rapidement la base initiale composée des variables logiques et réduire le nombre d'infaisabilités.
- PRIMAL : cette procédure calcule l'optimum du problème par la méthode du simplexe, forme produit de l'inverse. Les réinversions de la base sont demandées automatiquement après un certain nombre d'itérations (fréquence FIV). Ce mode de résolution s'applique à tous les problèmes qui ne font pas appel à des extensions de la programmation linéaire; dans ce cas, les possibilités décrites précédemment (INVOUT, SAVE, ...) et les analyses postoptimales sont valides.

3.2. Les extensions

Plusieurs possibilités sont incluses dans le système complet, soit pour augmenter la puissance de la programmation linéaire (programme séparable, programmation en nombres entiers), soit pour accroître les performances (techniques de décomposition, algorithme des variables bornées généralisées, algorithme de transport).

Les analyses postoptimales ne sont alors généralement pas prévues dans le système, et les possibilités de point de reprise, quand elles existent, sont dédicacées à un type particulier.

3.3. Analyses postoptimales

3.3.1. Analyse de sensibilité

Il en existe plusieurs types qui portent par exemple sur les coefficients de la fonction économique , sur les éléments du second membre ou sur les éléments de la matrice. Nous nous intéressons particulièrement à l'analyse qui concerne le second membre.
En généralisant la formule donnée dans la première partie (Chap. V, §6.2.) pour tenir compte de tous les types de variables, on veut calculer les bornes de variation, pour chaque élément indépendamment, qui garde la base actuelle inchangée. Le problème ne prend son sens que pour les variables hors base. Pour une étude plus détaillée, on peut se référer à Orchard-Hays [13] p. 142.

Les sorties de ce verbe, RNGRHS, peuvent être limitées à un sous-ensemble de variables qui lui est propre et qui est donc totalement indépendant du DELIMIT set. Toutes les impressions ont lieu sur SO.

3.3.2. Paramétrisation

Elle existe également pour différentes composantes du problème. Celle qui nous intéresse principalement concerne aussi le second membre : PARRHS.

Dans cette technique, c'est le vecteur tout entier qui est perturbé et non plus chaque élément en particulier. Elle consiste à faire varier le paramètre Θ positif dans $\min c.x$

$$\text{s.c. } A.x = b + \theta.\Delta b$$

$$x \geqslant 0$$

avec RHS = b, CRHS = Δb et THETA = Θ

Dans le MPS, il existe plusieurs modes de fonctionnement :

- au départ : a) si CRHSW = ON : THETA part avec sa valeur actuelle
  b) si CRHSW = OFF : THETA part de zéro.

- à la terminaison : on a toujours CRHSW = ON et THETA $\leqslant$ PARMAX

  a) THETA peut croître indéfiniment sans entraîner de changement de base; si PARMAX $< \infty$: THETA = PARMAX
  si PARMAX = ∞ : le système effectue la demande NOMAX

  b) THETA atteint PARMAX

  c) THETA $<$ PARMAX et il n'y a plus de changement possible
  si PARMAX $< \infty$: le système effectue la demande PREMAX
  si PARMAX = ∞ : terminaison normale et THETA garde sa valeur.

°

° °

CHAPITRE II : SOLUTIONS ADOPTEES

Après avoir présenté les possibilités du MPS, nous allons examiner les solutions que l'on peut apporter aux problèmes rencontrés dans l'algorithme. L'accent est mis sur chaque cas individuellement sans s'attacher à l'intégration générale de tous ces éléments, ce qui sera l'objet de la troisième partie du mémoire.

1. OPTIMISATION DES FONCTIONS ECONOMIQUES

1.1. Choix de l'algorithme

Etant donné que plusieurs techniques telles que la paramétrisation, l'analyse de sensibilité et les sorties de la matrice mise à jour seront utilisées dans l'algorithme, les méthodes de décomposition, de transport, ..., n'ont pas été retenues. Le calcul de l'optimum emploiera l'algorithme primal du simplex, par une procédure telle qu'elle a été décrite dans le chapitre précédent.

1.2. Optimum de plusieurs fonctions

Le MPS offre la possibilité et les outils pour optimiser plusieurs fonctions ; ce qui peut s'écrire :

```
SET  OBJ =< nom 1 > , SCALE = < facteur 1 >
PRIMAL
< impressions >
SET  OBJ =< nom 2 >, SCALE = < facteur 2 >
PRIMAL
etc...
```

Ceci oblige l'utilisateur à introduire une séquence d'instructions de ce type pour la résolution de chaque problème. Cette solution nuit à l'intégration et à la simplicité de l'interface. C'est pourquoi, nous avons utilisé une autre possibilité.

Parmi les paramètres que l'on peut passer aux routines-utilisateur, figurent OBJ et SCALE. Les optimisations successives seront sous la responsabilité de ces routines.

L'interface devra donc posséder une table des noms des fonctions à optimiser, avec les facteurs d'échelle, ainsi que les KJ que le système leur a affectés.

1.3. Construction de la matrice de paiement

Elle est construite progesssivement : chaque colonne étant garnie après l'optimisation de la fonction correspondante. Pour mettre ces valeurs à la disposition de l'interface, utiliser l'impression sur fichier, c'est-à-dire, en format carte, tout en limitant le nombre de sorties.
Le DELIMIT set courant doit donc reprendre toutes les fonctions économiques du problème.

2. CALCUL DES COEFFICIENTS DE POIDS

En plus des valeurs de la matrice de paiement, les coefficients des différentes fonctions économiques interviennent dans le calcul des poids. Nous avons déjà mentionné que l'interface prendra en charge la matrice du problème à partir de l'état de matrice en format interne.
La procédure ATOI permet de retraduire la matrice en format externe fixe, lisible facilement, puisqu'il s'agit d'images-cartes.
Lors de la lecture de ce fichier, d'autres informations nécessaires à l'interface peuvent être récoltées.

Toutes les variables sont rangées par ordre de KJ croissant, le premier bloc étant, d'après les règles associées aux KJ, celui des variables logiques. Pour se dresser une table de toutes les fonctions économiques du problèmes, il suffit donc de regarder, pour chaque variable logique, si son type est (FREE), et de noter alors le KJ correspondant, donné par un compteur qui est incrémenté à chaque nouvelle variable, quel que soit son type.

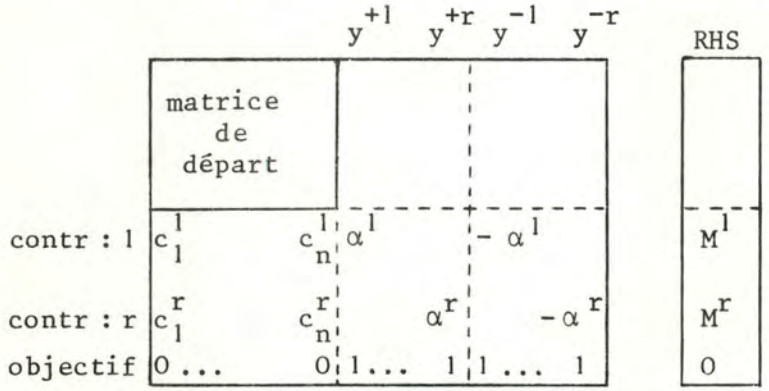
En continuant le balayage, on arrive sur la partie des éléments de la matrice. Pour chaque élément, il faut regarder s'il appartient à une des lignes identifiées dans la première étape, et calculer, le cas échéant, $\sum_{j=1}^{n} (C_j^i)^2$ pour le i qui a été identifié.

Pour terminer, compter le nombre de seconds membres et on a ainsi le nombre total de vecteurs dans le problème.

Une seule passe est nécessaire pour obtenir tous ces renseignements. Comme on y recueille, entre autres, le nom et le KJ des objectifs, la lecture du fichier doit se faire avant les optimisations; tandis que le calcul complet des coefficients de poids faisant intervenir la matrice de paiement, a lieu après. Ils seront donc calculés en deux étapes : avant et après les optimisations.

3. AJOUTER DES LIGNES ET DES COLONNES

Etant donné le genre de manipulations à faire, la seule possibilité à retenir est REVISE.
Schématiquement, les vecteurs peuvent se représenter sur le tableau :



Les différentes créations peuvent donc se décomposer :

a) déclarer contr:i comme combinaison linéaire de $f^i(x)$ $\quad i = 1, 2, \ldots, r$

$$\text{contr:i} = A \star f^i(x) + B \quad \text{avec } A = 1 \text{ et } B = 0$$

ceci introduit simultanément tous les coefficients C^i_j $\quad j = 1, 2, \ldots, n$ dans la ligne identifiée par contr:i.

b) déclarer les variables y^{+i} et y^{-i} en introduisant les coefficients α^i dans la contrainte correspondante, $i = 1, 2, \ldots, r$

c) déclarer l'objectif en introduisant les coefficients 1 des variables y^{+i}, y^{-i} $\quad i = 1, 2, \ldots, r$

d) modifier le RHS de contr:i par la valeur M^i, $i = 1, 2, \ldots, r$

L'ordre de ces opérations permet bien de construire la matrice étendue, mais il résulte d'une restriction MPS que les lignes combinaison linéaire ne sont créées qu'en fin de la procédure REVISE, ce qui nous oblige à grouper les

révisions en deux étapes : d'abord le point a) seul, puis les trois derniers points.

Le seul ennui de cette méthode est qu'il y a création de deux nouvelles matrices en format interne, ce qui peut causer des problèmes d'encombrement du fichier-problème. Ce point sera examiné avec les mécanismes de point de reprise qui utilisent également le fichier-problème pour sauver des informations.

Pour la modification des éléments du RHS, il importe d'en connaître le nom lors de la génération du fichier-MPS de révision : il constitue un nouveau paramètre à communiquer à l'interface.

4. MODIFICATION DE NIVEAUX

Changer un élément du second membre est une manipulation permise par MODIFY. Elle aura seulement un caractère temporaire et elle crée un fichier de travail incohérent avec la matrice en format interne, ce qui empêchera d'utiliser des points de reprise une fois que le problème aura été modifié. Mais comme nous avons cherché, dans l'algorithme, à donner le plus de facilités possibles pour le choix des relaxations des cibles, la lourdeur au point de vue temps et espace entraînée par l'utilisation répétition de REVISE est excessive, ce qui nous a fait adopter les modifications.

Une fois que le problème a été modifié, il suffit d'appeler PRIMAL pour obtenir la solution optimale et, au niveau du fichier de travail, cette modification est alors "oubliée".

5. BORNES DE VARIATION DES NIVEAUX DE SATISFACTION

Ces bornes sont données par l'analyse de sensibilité RNGRHS. Les seules lignes concernées sont les contraintes appelées contr:i, i = 1, 2,...,r, et qui ont été générées par l'interface. On peut donc constituer facilement un fichier contenant leur nom et qui servira à limiter les sorties. Ce sous-ensemble est propre à RNGRHS et n'a aucune interférence avec le DELIMIT set.

6. COEFFICIENTS MIS A JOUR

Les coefficients nécessaires, en se référant à la première partie, Chap. 4, § 6.2., où ils sont notés q^{i}_{m+1} i,l = 1, 2, ..., r, sont les coefficients correspondants :

- pour les variables en base, aux variables logiques de $f^{i}(x)$, i = 1, 2, ...,r

- pour les variables hors base, aux variables logiques des contraintes contr:i, i = 1, 2, ..., r.

Il y a deux moyens d'accéder à ces coefficients : INVOUT et ROWOUT; ils diffèrent par l'arrangement lignes/colonnes. Ce qui va permettre de choisir, c'est le DELIMIT set; on va le prendre égal à celui qui est nécessaire pour les valeurs des fonctions économiques. On garde donc le verbe INVOUT.

7. LES VECTEURS DE PERTURBATION

Dans l'étude de l'algorithme, nous avons donné deux possibilités d'amélioration globale qui dépendent du vecteur de perturbation choisi.

7.1. Le vecteur fixe

Il s'agit du vecteur (-1, -1, ..., -1)'. Comme il ne doit être généré qu'une seule fois, on le fera lors de la deuxième étape de révision : les seuls coefficients non nuls valent - 1 et sont situés sur les lignes contr:i , i = 1, 2, ..., r.

7.2. Les vecteurs variables

Comme on en génère un à chaque passage de cette étape, il faudra utiliser une procédure MODIFY. D'autre part, le nom devant être évidemment différent pour tous les vecteurs, le changement du paramètre CRHS ne pourra se faire que par l'interface, ce qui implique de connaître en plus du nom de ce vecteur, son KJ qui est égal au nombre courant de vecteurs plus un.

Les coefficients sont calculés à partir des valeurs actuelles des cibles et des valeurs obtenues, par RNGRHS, ce qui a déjà été décrit.

8. AMELIORATION GLOBALE

Il s'agit d'une problème paramétrique où THETA varie à partir de zéro et dont on ne connaît pas a priori de limite finie; on est donc forcé de prendre PARMAX égal à l'infini. La condition d'arrêt est donnée par le fait que la distance entre la cible et le compromis doit rester nulle.
Comme f (C) est compact , THETA aura certainement une valeur bornée, ce qui correspond à une terminaison normale de la paramétrisation.

Comment garder la distance nulle ? Les seules variables qui interviennent dans l'expression de la distance sont les variables y^{+i}, y^{-i}, $i = 1, 2, \ldots, r$. Lors de l'entrée dans cette étape, elles sont à un niveau nul et il faut les forcer à garder ce niveau, ce qui se fera par la procédure FLAGOUT. Le sous-ensemble de ces vecteurs aura été défini précédemment dans un fichier-MPS.

L'analyse du paramètre THETA se fera dans l'interface, ce qui implique qu'il fera également partie des arguments qui seront transmis.

9. SOLUTION APRES PARAMETRISATION

Le MPS offre la possibilité de continuer à travailler en mode composé pour le RHS. Cependant, cette facilité supprime toute itération supplémentaire sur cette étape. C'est pourquoi, il faut adopter une modification du RHS. Les seuls éléments à changer sont les cibles. L'interface connaît leur ancienne valeur, il connaît la valeur de THETA qui est passée comme argument, et il suffit d'avoir retenu les coefficients du vecteur de perturbation.

10. CONCLUSIONS

Nous venons de présenter les réponses apportées par MPS aux problèmes posés par l'algorithme. Excepté pour le DELIMIT set, l'accent n'a pas été mis sur la cohérence et l'intégration de ces solutions. Nous avons plusieurs fois mentionné l'existence d'un interface chargé de générer des fichiers, d'actualiser les valeurs des paramètres et d'orienter le déroulement de l'algorithme. Il fait l'objet de la partie suivante du travail.

Pour ce qui est des solutions mêmes, une caractéristique du MPS a fortement influencé les techniques utilisées : il s'agit de la procédure MODIFY. Tous les changements à apporter aux éléments du second membre utilisent cette procédure, que ce soit après une perturbation ou après une amélioration globale. Ensuite, l'algorithme PRIMAL rétablit l'optimum.

Si le passage du second membre composé à un nouveau second membre qui reprend la combinaison linéaire avait été plus simple, il aurait été possible d'utiliser plutôt des paramétrisations. Dans le cas de l'amélioration globale, c'est évident. Pour les relaxations, la procédure est également assez simple.

Soit M^i à changer en $M^i + R^i$, résoudre le problème paramétrique :

$$\min z$$

$$\text{s.c. } f^j(x) + \alpha^j . y^{+j} - \alpha^j . y^{-j} = M^j + \Theta \, \delta_i^j$$

$$x \in C, \; y^{+j} \geqslant 0, \; y^{-j} \geqslant 0, \; j = 1, 2, \ldots, r.$$

en fixant PARMAX = R^i

On a donc ainsi une condition d'arrêt pour la paramétrisation.

Les vecteurs (δ_i^j) sont générés une seule fois, lors des révisions, par exemple.

° ° °

TROISIEME PARTIE :

L'INTERFACE ET SON LANGAGE

INTRODUCTION

Dans la première partie, nous avons présenté une démarche particulière pour la résolution de problèmes linéaires à plusieurs critères. Ensuite, nous avons vu les principes de solution qu'offrait un software de programmation linéaire. Il nous reste à décrire l'interface et son langage qui permettront à un utilisateur de soumettre directement son problème multicritère et de construire interactivement le meilleur compromis à partir des techniques qui ont été étudiées.

Le premier chapitre expose le langage, d'abord à partir des fonctions qu'il supporte, ensuite en détaillant les différentes commandes.

Nous expliquons dans le deuxième chapitre, les principes adoptés dans l'interface : fichiers, structure et accès au terminal.

Nous terminons en intégrant tous les éléments de solution qui ont déjà été décrits et qui réalisent l'implémentation complète de l'algorithme.

o

o o

CHAPITRE I : SPECIFICATIONS DU LANGAGE

1. ORGANIGRAMME DE FONCTIONNEMENT

L'organigramme général est situé à la page suivante. Les traits pleins correspondent aux étapes de dialogue tandis que les traits pointillés reprennent les fonctions de l'interface. L'organigramme résume, de manière fonctionnelle, l'algorithme qui a été étudié.

2. LES ETAPES DE DIALOGUE

Le dialogue avec l'utilisateur est établi à quatre étapes distinctes.

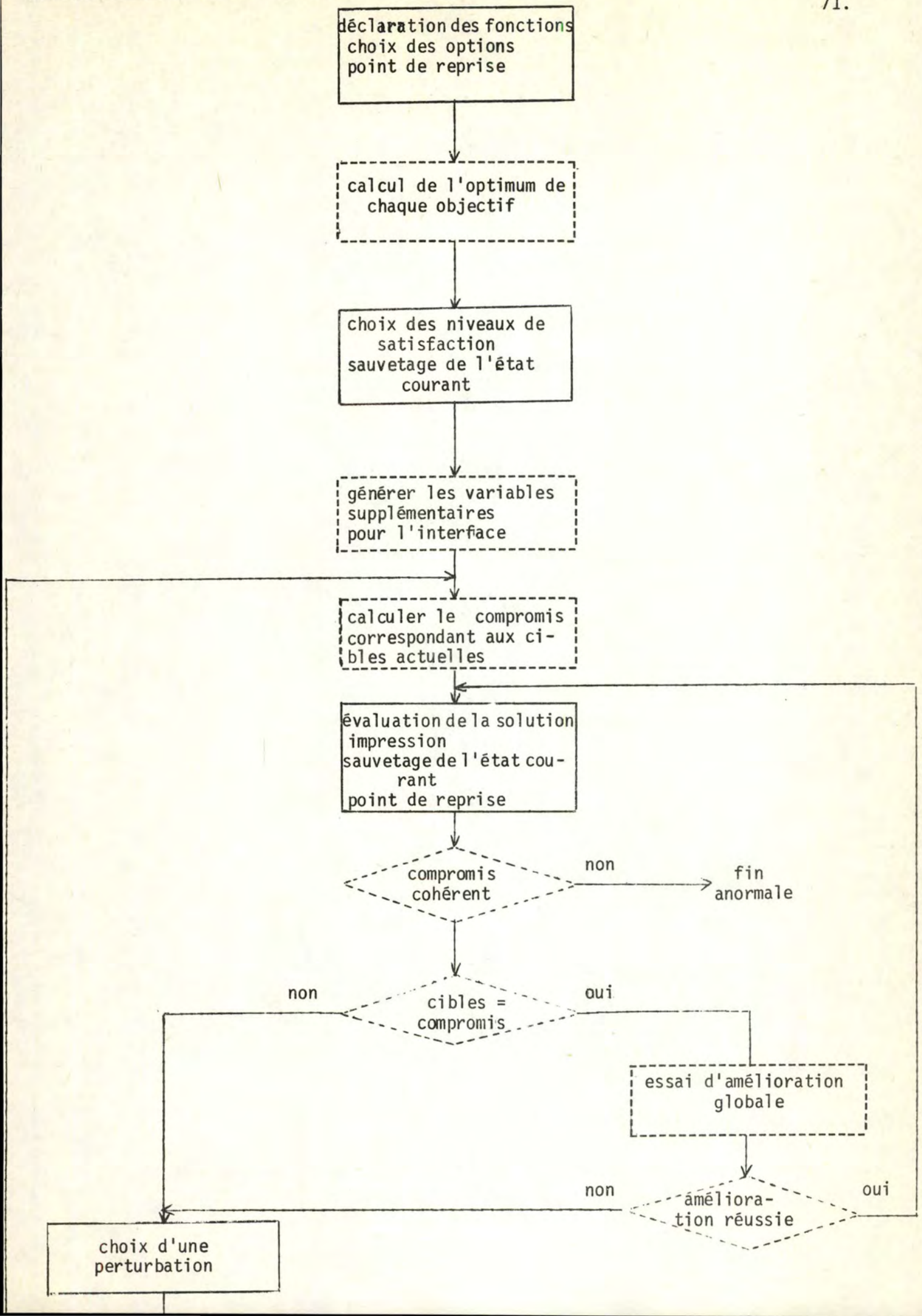
2.1 L'étape de déclaration

Elle sert à choisir les options de pondération qui ont été mentionnées lors de l'étude théorique. Nous avons proposé des poids pour normaliser les variables d'écart des différents objectifs. Ces poids peuvent être ramenés entre 0 et 1 (ce qui correspond à STEM [2]); ou bien le modèle n'est pas pondéré.

L'optimisation de plusieurs objectifs nécessite de changer les valeurs des arguments OBJ et SCALE de l'ACL. Pour atteindre une meilleure intégration, ce travail est réalisé par l'interface qui doit avoir construit les tables correspondantes. Implicitement, toutes les lignes (FREE) sont prises à minimiser ce qui est possible par le balayage du fichier ATOI. Si l'utilisateur désire retenir seulement un sous-ensemble de fonctions économiques, ou encore leur ajouter des facteurs d'échelle, il faut en faire explicitement la déclaration au cours de cette étape.

En dernier lieu, cette étape sert aussi à mentionner un point de reprise; les mécanismes seront détaillés plus loin.

Organigramme de fonctionnement



2.2. L'étape de choix des niveaux de satisfaction

Cette étape sera souvent appelée plus brièvement "étape 3", par rapport à l'algorithme donné dans la première partie.

La matrice de paiement, dont les avantages ont été soulignés, est maintenant construite et peut être consultée par l'utilisateur, ce qui lui permet d'introduire des cibles pour les objectifs.

Le travail qui précède cette étape (optimisation des fonctions principalement) ne demande pas l'intervention de l'utilisateur, il est important au point de vue temps d'exécution, ce qui justifie de pouvoir sauver l'état atteint à cette étape en vue de reprises ultérieures qui conservent les mêmes options et la même matrice du problème.

2.3. L'étape de présentation du compromis

Toujours par référence à l'algorithme, elle sera appelée "étape 6".

Le compromis courant est soumis à l'apprécation de l'utilisateur.

Dans le but de rendre utile l'étude du voisinage d'un compromis, la solution complète, c'est-à-dire les variables logiques et structurelles, peut être imprimée sur les listings du MPS.

Tant qu'aucune modification n'a été introduite, l'état courant peut également être sauvé pour une étude ultérieure. On vérifie bien ainsi les règles de cohérence entre le fichier de travail et le fichier-problème correspondant.

2.4. L'étape d'étude du voisinage du compromis actuel

L'utilisateur introduit une perturbation au compromis en changeant la valeur d'un des niveaux de satisfaction. Cette fonction se découpe en deux actions :

- indiquer le niveau à perturber; l'interface fournit alors les bornes correspondant à ce niveau.
- donner la valeur de la perturbation; l'interface imprime les valeurs attendues pour le nouveau compromis. Ces valeurs seront exactes à l'intérieur des bornes qui ont été données.

3. DESCRIPTION DES COMMANDES

3.1. Dans l'ACL

Toutes les fonctions qui ont été décrites dans la deuxième partie sont rassemblées en une seule macro qui assure également la logique de l'interface.

Les arguments qui sont nécessaires sont :

- la matrice en format interne : < fichier-problème >. Cela correspond à la convention qui a été adoptée au sujet des données prises en charge par l'interface. En plus de la diversité de construction de ce format, il est possible de le différer de l'exécution même de l'algorithme.
- l'unité sur laquelle se trouve cette matrice : < device >. Ceci pour utiliser les possibilités offertes par le MPS en ce qui concerne les fichiers à lire seulement, ou à lire et écrire. Ce file-code supporte également les sauvetages complets et le résultat des révisions; il sera expliqué en détail quand on parlera des fichiers de l'interface.
- le second membre : < rhs >. Le MPS permet d'avoir plusieurs seconds membres dans une même matrice et l'algorithme ne travaille que sur un seul à la fois.
- une indication sur le désir qu'a l'utilisateur d'imprimer la solution complète après l'optimisation de chacune des fonctions économiques : < impression >
- une indication sur le type de solution initiale fournie par l'utilisateur : < solution de départ >; ce qui peut être, soit rien, soit une base, soit une solution complète.
- le fichier sur lequel se trouve éventuellement la solution précédente < fichier de départ >. Il est omis quand il n'y a pas de solution.

Il suffit donc d'appeler la macro

GPSTEM < fichier-problème >; < device >; < rhs >; < impression >;
< solution de départ >; < fichier de départ >

où < device > ::= PT | XP
< impression > ::= PRINT | NONE
< solution de départ > ::= NONE | BASIS | SAVEWF

3.2. Dans l'interface

Nous décrivons ici les commandes qui sont proposées pour les étapes de dialogue de l'algorithme. Elles reprennent les fonctions qui sont apparues lors de l'étude du déroulement de l'algorithme, ainsi que les facilités ajoutées pour la souplesse de l'exécution. Elles sont relativement indépendantes, quant à leur fonction générale, des solutions adoptées dans cet interface, et pourraient donc servir également pour l'implémentation de cet algorithme sur un autre software de programmation linéaire.

Comme il s'agit d'un algorithme interactif, l'unité de programmation est la commande. Elles sont composées d'un verbe et d'une suite de phrases, éventuellement nulle. Dans la syntaxe des commandes :

- < nom d'objectif > correspond à l'identificateur d'une des fonctions économiques du problème. Suivant les règles du MPS, il s'agit d'un nom composé de trois parties de 6 caractères maximum, et séparées par :
- < valeur > nombre flottant respectant les conventions FORTRAN et d'une longueur maximum de 15 caractères.

3.2.1. Déclaration des objectifs

syntaxe : < commande DECLARE > ::= DECLARE < suite d'objectifs >
< suite d'objectifs > ::= < objectif > | < objectif > , < suite d'objectifs >
< objectif > ::= < nom d'objectif > < facteur d'échelle >
< facteur d'échelle > ::= < vide > | (SCALE = < valeur >)

sémantique :

Cette commande permet de déclarer les fonctions économiques à prendre en compte dans le problème, avec éventuellement un facteur d'échelle.

Elle n'est donc valide qu'à l'étape de déclaration.

Pour le MPS, cette commande n'est obligatoire que :

- si l'on veut limiter le nombre de fonctions économiques présentes dans la matrice de départ
- s'il faut déclarer un facteur d'échelle pour l'une des fonctions.

C'est toujours le dernier facteur d'échelle introduit qui est retenu pour un objectif, et s'il est omis, il vaut 1.

En cas d'introduction erronée d'un nom, il faut annuler toutes les déclarations et recommencer (voir commande SET).

3.2.2. Affichage des valeurs relatives aux objectifs

syntaxe : < commande DISPLAY > ::= DISPLAY < suite de vecteurs >
< suite de vecteurs > ::= < vecteur > | < vecteur > , < suite de vecteurs >
< vecteur > ::= NAMES | LEVEL | PAYOFF | WEIGHT | RESULT | SLACK

sémantique :

Affichage des valeurs correspondantes

- NAMES : les fonctions économiques et les facteurs d'échelle
- LEVEL : les niveaux de satisfaction
- PAYOFF : la matrice de paiement
- WEIGHT : les facteurs de poids du modèle
- RESULT : le compromis courant
- SLACK : les variables d'écart

Y:PLUS > 0 si la cible est moins bonne que la valeur de l'objectif
Y:MINUS > 0 si la cible est meilleure
"meilleur" signifie "cible < objectif" pour la minimisation
"cible > objectif" pour la maximisation

Cette commande est valide à toutes les étapes, toutefois, elle n'a de sens que pour les valeurs déjà initialisées.

3.2.3. Terminaison

syntaxe : < commande ENDGP > ::= ENDGP

sémantique :

Elle clôture l'exécution de l'algorithme et imprime la solution complète.

3.2.4. Continuation

syntaxe : < commande EXECUTE > ::= EXECUTE

sémantique :

Elle relance le déroulement de l'algorithme, en effectuant lors de l'étape de perturbation, les modifications demandées par le dernier niveau proposé et la dernière valeur entrée.

3.2.5. Impression du format des commandes

syntaxe : < commande HELP > ::= HELP | HELP < suite de verbes >
< suite de verbes > ::= < verbe > | < verbe >, < suite de verbes >
< verbe > ::= DECLARE | DISPLAY | ENDGP | EXECUTE |
HELP | MODIFY | OUTPUT | RESTORE |
RNGLVL | SAVE | SET | SETLVL | STATUS

sémantique :

Elle imprime un rappel du format des différentes commandes.
Si aucune n'est spécifiée, la liste complète est imprimée.

3.2.6. Perturbation d'un niveau de satisfaction

syntaxe : < commande MODIFY >::= MODIFY < valeur >

sémantique :

Le compromis attendu après une modification correspondant à < valeur > de la cible associée à l'objectif qui a été introduit précédemment par RNGLVL, est imprimé. Ce compromis est exact à l'intérieur des bornes fournies, sinon il s'agit d'une approximation linéaire du compromis réel.

Cette commande n'est évidemment valide qu'au cours de l'étape de perturbation. La valeur introduite est retenue pour effectuer la modification définitive à la fin de cette étape, c'est-à-dire lors de l'introduction de la commande EXECUTE.

3.2.7. Impression de la solution

syntaxe : < commande OUTPUT > ::= OUTPUT

sémantique :

Au cours de l'étape de présentation du compromis, la solution complète peut être obtenue sur le fichier auxiliaire d'impression XO. C'est la même procédure d'impression que pour ENDGP, mais sans clôturer l'algorithme.

3.2.8. Point de reprise

syntaxe : < commande RESTORE > ::= RESTORE

sémantique :

Réinstallation de la solution complète du MPS et du fichier de travail de l'interface, puis positionnement à l'étape correspondante.

Le principe des sauvetages et des points de reprise sera exposé dans un paragraphe particulier.

Cette commande peut s'utiliser dans deux contextes :

- au cours de l'étape de déclaration pour reconfigurer l'interface, soit à l'étape 3, soit à l'étape 6.
- au cours de l'étape 6, ce qui permet de recommencer l'étude du voisinage du compromis en effaçant toutes les modifications qui ont déjà été introduites.

3.2.9. Choix d'une cible à perturber

syntaxe : < commande RNGLVL > ::= RNGLVL < nom d'objectif >

sémantique :

Lors de l'étape d'étude du voisinage du compromis actuel, cette commande sélectionne le niveau à perturber. C'est à ce moment que sont données les bornes de variation du niveau, qui laissent la base inchangée. A l'intérieur de ces bornes, la solution estimée par la commande MODIFY est donc exacte.

Ces commandes n'impliquent aucune modification dans les fichiers du MPS, il s'agit d'approximations réalisées par l'interface; ce qui rend cette étape très peu coûteuse, tout en offrant de nombreuses possibilités pour guider le choix. Les différents théorèmes vus dans la première partie peuvent également intervenir dans le choix des modifications.

3.2.10. Sauvetage de l'état courant

syntaxe : < commande SAVE > ::= SAVE

sémantique :

Aux étapes 3 et 6, cette commande crée un fichier solution appelé, suivant le cas, GPSTEM : SAVE : $\begin{pmatrix}3\\6\end{pmatrix}$. Parallèlement, l'interface sauve ses tableaux et ses variables de contrôle. Suivant les règles du MPS, cette commande n'a donc plus de sens dès que la première modification a été introduite à l'étape 6, que ce soit par l'étape d'amélioration globale ou de perturbation des niveaux.

3.2.11. Choix des options

syntaxe : < commande SET > ::= SET < suite d'options >
< suite d'options > ::= < option > | < option >, < suite d'options >
< option > ::= < mot clé > = < valeur booléenne > | NOBJ = 0
< mot clé > ::= AUTOSCALE | RANGE | SAME | NOSO
< valeur booléenne > ::= ON | OFF

sémantique :

- AUTOSCALE : Choix d'un modèle pondéré ou non. Il s'agit des poids proposés dans la première partie. Sa valeur implicite est ON.

- RANGE : ces poids sont ramenés entre 0 et 1 (méthode STEM).
  Sa valeur implicite est OFF.

- SAME : choix entre les vecteurs pour l'étape d'amélioration globale, soit la même diminution de tous les objectifs, soit une diminution proportionnelle à leur variation qui n'entraîne pas de changement de base.
  Sa valeur implicite est OFF.

- NOSO : commande la suppression des sorties standards du MPS concernant l'évolution de l'exécution : les verbes, les itérations primales, ...
  Sa valeur implicite est OFF.

- NOBJ = 0 : cette phrase permet d'initialiser à nouveau la table des noms des fonctions en cas d'introduction erronée.

La validité des différentes phrases dépend évidemment des paramètres qu'elles gouvernent :
étape de déclaration : AUTOSCALE, RANGE, NOBJ
toutes les étapes : SAME, NOSO

3.2.12. Choix des niveaux de satisfaction

syntaxe : < commande SETLVL > ::= SETLVL < suite de niveaux >
< suite de niveaux > ::= < niveau > | < niveau > , < suite de niveaux >
< niveau > ::= < nom d'objectif > = < valeur >

sémantique :

Elle permet d'introduire les niveaux de satisfaction pour chaque objectif, ce qui fixe l'étape de validité à l'étape 3.

Implicitement la valeur de la cible est égale à l'optimum de l'objectif correspondant.

De plus, en vertu du théorème 3 de la première partie, il est inutile d'introduire une cible meilleure que l'optimum.

3.2.13. Affichage des contrôles de l'interface

syntaxe : < commande STATUS > ::= STATUS | STATUS < suite de contrôles >
< suite de contrôles > ::= < contrôle > | < contrôle > , < suite de contrôles >
< contrôle > ::= TIME | NOBJ | KJ | OPTION | PRTLVL

sémantique :

Si aucune des phrases n'est donnée, il y a affichage de toutes les valeurs.

- TIME : temps CPU utilisé pour cette exécution de l'algorithme.

- NOBJ : nombre de fonctions économiques

- KJ : nombre total de variables (logiques, structurelles et RHS)
  Ce nombre évolue après la génération des révisions et après les étapes d'amélioration globale, si SAME = OFF.
- OPTION : options utilisées pour cette exécution
- PRTLVL : uniquement pendant l'étape d'étude du voisinage, donne le nom de l'objectif dont le niveau est perturbé.

4. LES SOLUTIONS ANTERIEURES ET LES POINTS DE REPRISE

Nous exposons ici le mécanisme des points de reprise en détaillant également les techniques utilisées pour implémenter ces mécanismes.
De ce fait, ce paragraphe est fortement orienté par le MPS.

4.1. Résultats antérieurs

Les premières optimisations concernent la matrice telle que l'utilisateur l'a donnée. Il en possède peut-être des résultats, soit sous forme de base optimale pour un des objectifs, soit sous forme de fichier de sauvetage complet.

La distinction a lieu lors de l'appel à l'interface dans l'ACL, et est réalisée par les deux derniers arguments qui sont passés à la macro GPSTEM.

La séquence d'instructions qui réalise la mise en place de l'une ou l'autre possibilité est :

```
PERFORM    SETUP
  #5         #6
...
```

où SETUP est l'étiquette de la sous-routine, niveau ACL, qui effectue la mise en place du fichier de travail et le cinquième argument est NONE, BASIS ou SAVEWF. Ces mots réservés ne sont que les appels à des macros élémentaires dédicacées à chacun des cas :

a)
```
MACRO    NONE
PERFORM  NEXT
ENDM
```
La routine identifiée par l'étiquette NEXT est une routine vide, composée uniquement du verbe NEXT. Ce qui correspond au cas où l'utilisateur n'introduit aucun résultat.

b)
```
MACRO    BASIS
LDBASIS  SOURCE = #1
ENDM
```
mise en place d'une base de départ

c) MACRO SAVEWF
RESTORE SOURCE = #1
ENDM
mise en place d'une solution complète

4.2. Points de reprise

La création des fichiers de sauvetage a lieu après les deux grandes étapes d'optimisation : le garnissage de la matrice de paiement, et le calcul du premier compromis. Pour respecter les règles du MPS concernant ces sauvetages, il faut toujours garder en parallèle deux fichiers-MPS : la matrice en format interne et la solution complète.

Lors de l'étape 3, le sauvetage consiste à créer un fichier GPSTEM : SAVE : 3 qui doit exister avec < fichier-problème > qui est la matrice en format interne, fournie par l'utilisateur.

Lors de l'étape 6, il s'agit du fichier GPSTEM : SAVE : 6 et de GPSTEM : CONV : 1 qui est le résultat de la deuxième révision générée par l'interface.

A cela, il faut ajouter que l'interface possède également un fichier de sauvetage pour conserver le nom des objectifs, la matrice de paiement, ..., et d'autres informations de gestion interne.

Le sauvetage comporte donc les opérations suivantes :

- dans l'interface : écrire toutes les informations reprenant l'état actuel, plus une indication pour l'étape atteinte : 3 ou 6.

- dans le MPS : création du fichier de sauvetage
SAVE IDENT = GPSTEM : SAVE : $\begin{pmatrix}3\\6\end{pmatrix}$
suivant le cas.

Pour la mise en place de la solution antérieure, on peut aussi distinguer :

- dans l'interface : lecture des informations sauvées; l'indication 3 ou 6 permet d'orienter la suite du déroulement. Il faut noter cependant que les anciennes valeurs concernant l'identification du terminal ou le pointeur dans le fichier des commandes ne sont plus valides et sont à remplacer par les valeurs courantes (voir Chap. II, 3. Les accès au terminal)

- dans le MPS : deux actions sont nécessaires : recréation du fichier de travail et chargement de la solution

SETUP SOURCE = { < fichier-problème > / GPSTEM : CONV : 1 }

RESTORE SOURCE = GPSTEM : SAVE : { 3 / 6 }

°

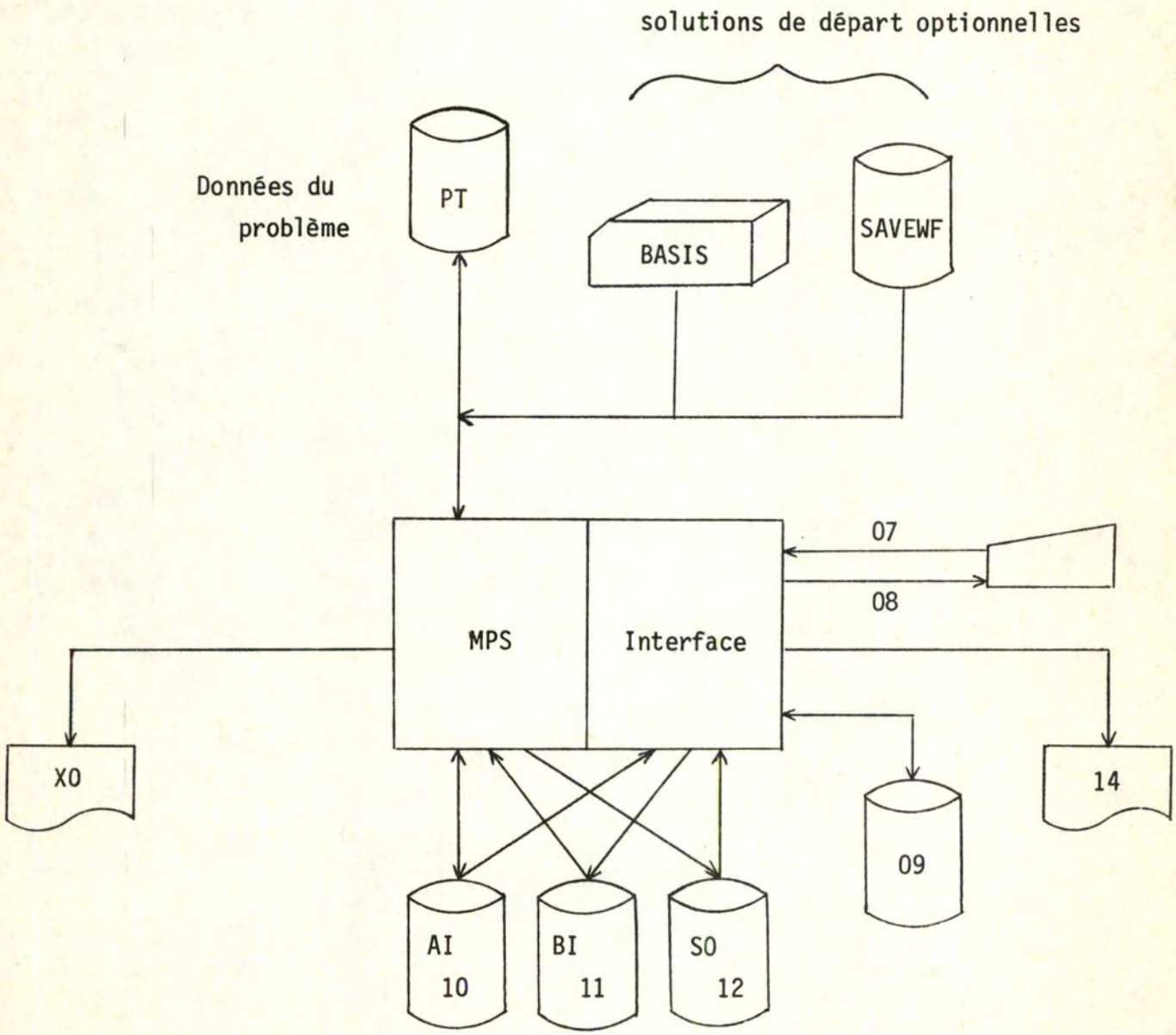
° °

CHAPITRE II : LES PRINCIPES DE L'IMPLÉMENTATION

1. LES FICHIERS

Le MPS et l'interface se partagent ou utilisent en propre des fichiers, niveau GCOS, pour le déroulement de l'algorithme. Nous ne reprenons dans cette description que ceux qui sont typiquement dédicacés à ce déroulement.

1.1. Graphe général



Les fichiers sont identifiés par leur file-code, excepté pour les solutions de départ optionnelles qui sont reprises sous le nom du paramètre de la macro GPSTEM. Ces solutions se trouvent sur un des autres file-codes disponibles, et adapté au type d'information.

1.2. PT

Les fonctions de ce fichier ont déjà été décrites lors de la présentation du MPS. Dans l'exécution de l'algorithme, il contiendra les deux types d'informations : matrices de format interne et solutions complètes.

En effet, l'interface va faire générer sur PT :

- GPSTEM : CONV : 0 matrice intermédiaire contenant les données initiales plus les lignes combinaison linéaire des objectifs;
- GPSTEM : CONV : 1 matrice qui contient toutes les lignes et les colonnes initiales et celles qui sont nécessaires pour l'algorithme;
- GPSTEM : SAVE : 3 solution complète obtenue au cours de l'étape de choix des niveaux de satisfaction;
- GPSTEM : SAVE : 6 solution complète de l'étape de présentation du compromis.

Ces deux fichiers n'existent évidement qu'à la demande de l'utilisateur.

La taille de PT peut devenir contraignante. Il est possible d'effectuer une extraction des fichiers intéressants (matrice en format interne et solution correspondante). Le MPS offre le moyen d'effectuer une telle sélection par l'ordre COPY. Par exemple, pour obtenir les fichiers nécessaires à l'étape 6, on peut utiliser :

```
COPY  SOURCE = GPSTEM : CONV : 1/XP, IDENT = GPSTEM : CONV : 1,
              TYPE = REVISE

COPY  SOURCE = GPSTEM : SAVE : 6/XP, IDENT = GPSTEM : SAVE : 6,
              TYPE = SAVE
```

où XP est l'ancien fichier-problème qui n'est utilisé qu'en lecture, les informations étant copiées sur le deuxième fichier-problème, déclaré avec le file-code PT.

1.3. AI-10

C'est le premier fichier de communication entre le MPS et l'interface. Le file-code AI est celui du MPS tandis que 10 est celui des routines FORTRAN. Il sert également de stockage intermédiaire de la base lors des révisions.

Il contient successivement :

- GPSTEM : ATOI, ce sont les données de l'utilisateur reconverties en format externe ;

- GPSTEM : ETAPE : 1, valeurs prises par les fonctions économiques. Il sert lors de la constitution de la matrice de paiement et lors du calcul d'un compromis;

- GPSTEM : BASIS, base du problème qui permet de gagner des itérations après les révisions.

Il est écrit uniquement par le MPS. Comme les durées de validité de chacun de ces fichiers ne sont jamais imbriquées et qu'ils sont temporaires, ils sont effacés avant chaque utilisation.

1.4. BI-11

C'est le deuxième fichier de communication, qui est écrit entièrement par l'interface. Il contient :

- GPSTEM : MASK : OBJ, fichier de masques du nom des fonctions économiques. Il est utilisé pour la création du DELIMIT set.

- GPSTEM : MASK : CONSTR, fichier de masques du nom des contraintes pour la norme L1. Il sert à délimiter le nombre de vecteurs pour l'analyse de sensibilité (verbe RNGRHS) et le nombre de colonnes de la matrice mise à jour (verbe INVOUT).

- GPSTEM : MASK : SLACKS, fichier de masques du nom des variables d'écart y^{+i}, y^{-i}, i = 1, 2, ..., r. Dans l'étape d'amélioration globale, il sert à empêcher ces variables d'entrer dans la base lors de la paramétrisation.

- GPSTEM : REVISE : FILE, fichier des révisions intermédiaires. Il introduit les contraintes pour la norme L1 comme combinaison linéaire de chacune des fonctions économiques.

- GPSTEM : REVISE : FILE1 , second fichier des révisions, il complète et met à jour la première étape.

- GPSTEM : MODIFY : FILE, fichier des modifications. Il contient, soit les changements résultant de l'étape d'étude du voisinage, soit un nouveau second membre pour l'étape d'amélioration globale, soit les changements impliqués par cette étape

Il n'est généré qu'au début de l'exécution et par la suite, il subit uniquement des mises à jour (pour GPSTEM : MODIFY : FILE, par exemple), ce qui permet d'avoir un nombre fixe de fichiers-MPS, et ainsi de déclarer aisément les noms respectifs dans la macro GPSTEM.

L'extension maximale est de 22 + 13* NOBJ cartes où NOBJ et le nombre de fonctions économiques pour cette exécution.

1.5. SO-12

Troisième fichier de communication, il s'agit en fait des impressions standard du MPS qui doivent être assignées à un fichier disque pour permettre à l'interface de lire "on-line" les informations qui y sont écrites : les verbes RNGRHS et INVOUT écrivent uniquement sur SO.

Son exploitation pose deux problèmes qui sont interdépendants. Pour garder la validité physique du fichier, il faut conserver la séquence des enregistrements physiques. Tout rebobinage effectué dans l'interface doit être suivi d'un repositionnement correct pour rendre la manipulation invisible au MPS qui ne connaît qu'un seul type d'action : écrire la ligne suivante. C'est pourquoi, les informations s'accumulent au cours du déroulement de l'algorithme et qu'il faut résoudre les points suivants :

- retrouver les informations.

On utilise à deux reprises des informations placées sur SO :

- dans l'étape d'étude du voisinage du compromis, la séquence des verbes est RNGRHS RLIST = GPSTEM : MASK : CONSTR/BI
  INVOUT RLIST = GPSTEM : MASK : CONSTR/BI
- dans l'étape d'amélioration globale
  RNGRHS RLIST = GPSTEM : MASK : CONSTR/BI

De plus, on sait identifier univoquement le début des états imprimés concernant le verbe RNGRHS.

La solution consiste donc :
- à avoir un compteur dans l'interface initialisé à zéro;
- à l'incrémenter de 1 à chacune des étapes qui viennent d'être citées;
- à rebobiner SO et à le relire jusqu'à l'état RNGRHS numéro n où n est la valeur courante du compteur.

- repositionner le fichier.

La dernière ligne écrite par le MPS avant l'intervention de l'interface est celle indiquant l'appel aux routines-utilisateur

CALL GPFOR/...

Une fois que les bonnes informations ont été enregistrées, il suffit de lire le fichier SO jusqu'au premier CALL suivant et toutes les manipulations restent ainsi invisibles pour le MPS.

L'utilisateur peut être intéressé par le suivi de l'exécution, ce qui nécessite l'impression du fichier SO. Cette opération, contrôlée par le paramètre NOSO de l'interface, n'est effectuée qu'à la clotûre de l'algorithme.

1.6. XO

C'est le fichier auxiliaire d'impression du MPS accessible par le verbe OUTPUT qui commande l'impression des résultats complets (variables logiques et structurelles).

Il contient les résultats :

- des optimisations successives si PRINT est spécifié dans la macro GPSTEM;
- demandés par OUTPUT (commande de l'interface);
- demandés par ENDGP

Toutes ces impressions sont effectuées par la routine, niveau ACL.

```
OUTP      OUTPUT    ALL,XO
          NEXT
```

Le paramètre ALL supprime temporairement le DELIMIT set, ce qui permet l'impression complète. Pour personnaliser les résultats, il suffit de modifier cette routine.

1.7. 07 et 08

Il s'agit des fichiers d'accès au terminal : 07 pour l'entrée des commandes et 08 pour les impressions.

Le mécanisme détaillé de ces accès sera expliqué dans un paragraphe particulier.

1.8. 09

Fichier propre à l'interface, il est utilisé pour sauver toutes ses informations. Ceci est nécessaire puisqu'aucune zone n'est rémanente dans les routines entre les appels du MPS.

Il est organisé en mode random et il possède deux enregistrements logiques : celui utilisé entre tous les appels, et celui, éventuellement utile, pour les mécanismes de points de reprise. La longueur d'un enregistrement logique est fixe et vaut 300 caractères.

1.9. 14

Il s'agit du fichier supportant la copie du fichier SO lors de la clôture de l'exécution de l'algorithme. Son affectation est inutile si NOSO = ON.

2. STRUCTURE GENERALE DE L'INTERFACE

2.1. La racine

Les routines qui composent l'interface sont organisées de manière hiérarchiques. Tout appel par le MPS à l'interface commence par dérouler des instructions dans la routine principale, MAIN, qui constitue la racine de cette structure. Elle a deux fonctions importantes :

- gestion du fichier de sauvetage de l'interface : 09

  Un enregistrement logique contient :

  - les informations relatives aux fonctions économiques (nombre, noms, KJ, poids, cibles, ...);
  - les options de fonctionnement;
  - les indications pour la gestion des accès au terminal ainsi que le compteur associé au fichier SO.

  Lors de l'entrée dans la routine, excepté pour le premier appel, l'enregistrement est lu pour reconfigurer les informations de l'interface. De même, à chaque fin d' exécution l'enregistrement complet est écrit pour le prochain appel.

Pour un sauvetage, l'état courant est également écrit sur l'enregistrement dédié à cette fonction et à la reprise, les valeurs actuelles sont remplacées par cet enregistrement. Il faut cependant conserver les informations qui concernent typiquement cette exécution (gestion du terminal et compteur pour SO)

- la sélection des routines particulières

Lors des étapes de dialogue, toutes les commandes sont lues à partir du même point de la routine MAIN quelle que soit l'étape. Un message est cependant envoyé pour déterminer le contexte du dialogue. Après la lecture, le verbe est analysé; si cette commande possède des phrases, une routine particulière est appelée et la suite de l'interprétation est exécutée par cette routine. Dans le cas de commandes sans phrases, l'interprétation complète a lieu dans la routine MAIN.

Lors des étapes de calcul, le processus est orienté vers la ou les routines dédicacées à cette étape.

De plus, dans le but de limiter l'encombrement mémoire qui provoquait des interférences avec le MPS, l'ensemble de l'interface possède sa propre structure d'overlay. Le segment racine est contitué :

- de la routine MAIN;
- des routines d'accès au terminal;
- des routines pour l'analyse lexicographique et pour les modifications d'adresse des arguments transmis (voir plus bas)

Avant d'initialiser un traitement particulier, la routine MAIN doit donc commander le chargement de la branche correspondante.

2.2. Le contrôle de séquence

Nous avons déjà vu plusieurs fois le problème du contrôle de séquence, soit lors des appels de l'interface, soit lors du retour dans l'ACL.

2.2.1. Dans l'interface

Le point d'entrée y est unique, c'est la routine MAIN, dont une des fonctions est d'orienter le processus. Ce qui implique d'avoir comme argument lors du passage MPS-interface, une indication de l'étape atteinte.

Lors de la lecture des commandes, l'envoi d'un message spécifiant le contexte, est fait dans MAIN, puis le reste de l'analyse de la commande est exécuté dans un module dédicacé. L'indicateur d'étape sera également transmis à ces routines qui nécessitent un contrôle de validité du contexte (DECLARE, SETLVL, ...)

2.2.2. Dans l'ACL

Au cours de l'étape d'amélioration globale, le paramètre THETA est testé dans l'interface et suivant sa valeur, l'algorithme est orienté ou non vers l'étape d'étude du voisinage. L'argument contrôlant la sélection des routines de l'interface aura également comme fonction d'orienter la séquence des verbes ACL. Il existe d'autrescas où la suite du déroulement est commandée par l'interface :

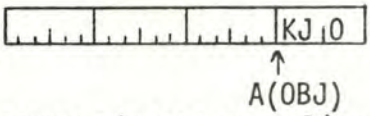
- les verbes du langage qui necessitent l'exécution d'instructions de l'ACL (SAVE, RESTORE, OUTPUT);
- la clôture de l'algorithme, soit demandée par ENDGP, soit due à une condition de terminaison anormale : $y^{+i} > 0$, par exemple.

2.3. Les arguments transmis

Nous reprenons ici les différents arguments dont la nécessité a été mise en évidence lors des paragraphes précédents, et qui sont transmis par le MPS aux routines de l'interface. L'instruction d'appel est

CALL GPFOR/USER1, OBJ, RHS, CRHS, THETA, SCALE

- USER1 : une des variables de communication du MPS, prenant des valeurs flottantes entre 0.0 et 100.0. Elle reprend les fonctions d'orientation de l'exécution entre les verbes ACL et les routines de l'interface.
- OBJ : nom de la fonction économique courante. Il est mis à jour par les routines dans l'étape d'optimisation de toutes les fonctions.
  Le format général des noms-arguments est le suivant :



Le MPS passe comme adresse pour l'argument OBJ, la valeur qui est notée A(OBJ) et qui correspond à l'adresse du mot qui contient le KJ dans le demi-mot de gauche. Le nom est placé dans les trois mots (3 fois 6 caractères) précédants le KJ.

La lecture et la modification des valeurs ont été réalisées dans l'interface par deux routines écrites en langage de base pour permettre les mo-

difications d'adresse. Il faut encore noter que dans le cas de la fonction économique , le KJ est placé dans les 17 bits de gauche seulement.

- RHS : nom du second membre courant. Il est lu par l'interface pour générer les fichiers de révision et de modification.
- CRHS : nom du second membre composé. Il est modifié par l'interface au cours de l'étape d'amélioration globale. Il contient alors, soit le nom du vecteur constant : GPSTEM : CRHS : ETAP8, soit le nom du nouveau vecteur qui vient d'être généré : GPSTEM : CRHS : n où n est le nombre d'itérations à cette étape.
- THETA : coefficient de la paramétrisation. Sa valeur est testée dans l'interface.
- SCALE : facteur d'échelle de la fonction économique courante. Il est positionné en même temps que OBJ à la valeur correspondante.

3. LES ACCES AU TERMINAL

3.1. Principes des accès

Les entrées du terminal amènent les commandes tandis que les sorties supportent les informations demandées par ces commandes, ainsi que les impressions résultant du déroulement de l'algorithme. Bien qu'une exécution soit essentiellement interactive, il peut être utile d'effectuer certaines étapes en batch pour ne pas bloquer inutilement un terminal suite aux longues phases de calcul; par exemple, soit le programme :

```
EXECUTE
DISPLAY  PAYOFF
SAVE
ENDGP
```

qui demande de prendre en compte toutes les fonctions économiques en les minimisant, puis qui imprime la matrice de paiement, sauve l'état atteint et clôture l'exécution.

L'interface doit donc supporter les deux types d'entrées-sorties : sur fichier pour les exécutions batch et sur terminal interactif. Le MPS étant dans tous les cas, un programme batch pour le système d'exploitation, ces deux modes demandent des traitements particuliers.

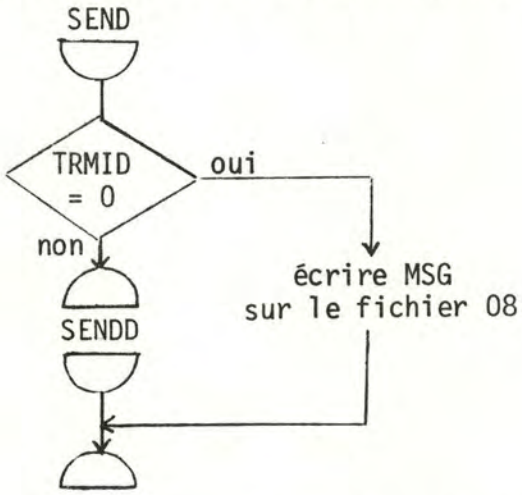
Pour banaliser ces traitements les entrées consistent à appeler la routine ACCEPT qui fournit la commande suivante dans la variable CARD, tandis que pour les sorties, il suffit d'appeler SEND qui écrit le message qui a été préalablement formatté dans la variable MSG. C'est à l'intérieur de ces deux routines que se fait la distinction de mode de fonctionnement.

3.2. Identification du mode

Implicitement, l'interface travaille avec le mode terminal-interactif Lors de la première entrée dans la routine MAIN, il y a appel au superviseur pour obtenir l'identification de l'éventuel terminal qui a demandé la connexion avec le programme. Après deux minutes maximum d'attente, l'interface considère que le terminal est simulé par les fichiers 07 et 08 pour tout le reste de l'exécution.

3.3. Mode terminal-interactif

En cas de réponse affirmative à l'étape précédente, le superviseur transmet le code du terminal concerné, sinon ce code garde la valeur zéro.
La routine SEND, par exemple, a donc la structure suivante :



TRMID est la variable qui contient l'identification, et SENDD est la routine, écrite en langage de base, qui envoie le message au terminal identifié par TRMID. Le principe est le même pour l'entrée des commandes.

3.4. Exécution batch

Dans ce cas, TRMID vaut zéro et les routines ACCEPT et SEND effectuent les entrées-sorties correspondantes.

Après la lecture de chaque commande, elle est de plus imprimée pour la clarté des listings de sortie. Un problème supplémentaire se pose. Nous avons déjà mentionné que l'interface n'a aucune zone rémanente entre les appels du MPS, en particulier pour la zone de description du fichier 07. Il faut donc tenir à jour un compteur d'instructions ayant demandé un retour au MPS et

sauter autant d'instructions à chaque nouvelle entrée dans l'interface.

En conséquence, le fichier 07 doit être sur un support séquentiel qui permet les lectures répétées des mêmes enregistrements (pas un lecteur de cartes par exemple).

°

° °

CHAPITRE III : ETUDE DE L'IMPLÉMENTATION

Dans ce chapitre, nous insistons principalement sur l'intégration de l'interface. La plupart des solutions ponctuelles ont déjà été mentionnées, soit lors de la présentation du MPS dans la deuxième partie, soit lors du chapitre précédent.

Nous détaillerons cependant les modifications qu'implique la prise en charge d'un facteur d'échelle pour les objectifs.

1. LES DECLARATIONS

Lors du premier appel de l'interface par le MPS, il y a identification du mode de fonctionnement (batch-interactif), envoi du message de bienvenue et branchement à la lecture des commandes. Si l'utilisateur emploie DECLARE, deux tables sont garnies, classées par ordre des déclarations : la table des noms et la table des facteurs d'échelle. Le principe des points de reprise a déjà été étudié.

A la fin de cette étape de dialogue, il y a retour au MPS pour créer la matrice en format ATOI. L'interface est de nouveau appelé pour une étape qui n'a pas de dialogue. Le balayage du fichier ATOI recouvre plusieurs fonctions :

- Le comptage de toutes les variables logiques, structurelles et RHS, qui se fait au fur et à mesure de la lecture.
- Dans la partie des variables logiques :
  - si l'utilisateur a déjà déclaré ses fonctions, il y a contrôle de validité de ces déclarations et réarrangement des deux tables par ordre KJ croissant. Parallèlement, la table KJ est aussi construite. Ceci correspond à l'option qui a été prise dans l'interface et qui veut que toutes les tables qui concernent les fonctions économiques soient classées suivant cet ordre. Le principal avantage est la concordance avec l'ordre du MPS, lors des impressions et lors de la lecture des solutions, de la matrice mise à jour et des résultats de l'analyse de sensibilité.

- si l'utilisateur n'a pas encore déclaré ses objectifs, les tables des noms et des KJ sont construites simultanément. Pour les facteurs d'échelle, la valeur implicite est 1.

Il y a deux cas de terminaison anormale au cours de cette étape :

- l'utilisateur a déclaré un nom de fonction qui n'existe pas dans le fichier ATOI;
- l'utilisateur n'a rien déclaré, et il existe trop de fonctions économiques dans ce fichier. Ce problème ne se pose pas avec la commande DECLARE; vu son caratère conversationnel, un avertissement est simplement envoyé.
  Actuellement, le nombre maximum de fonctions économiques, appelé NMAX, vaut 8. C'est un paramètre de compilation. Il ne doit être confondu avec NOBJ qui est une variable de l'exécution et qui représente le nombre courant d'objectifs pour cette exécution; en reprenant les notations adoptées jusque maintenant, NOBJ = r.

- Dans la partie des éléments de la matrice :

  - si AUTOSCALE = OFF, tous les poids valent 1, et la lecture continue simplement pour le comptage des vecteurs jusqu'à la partie suivante;
  - sinon, chaque élément est examiné pour savoir s'il appartient à une des fonctions. Il suffit pour cela de lire le nom de la ligne de cet élément. En cas de succès, on calcule $\sum_{j=1}^{n} (C_j^i)^2$ pour le i correspondant.

- Dans la partie des RHS, on achève de compter le nombre total de vecteurs.

Cette étape s'achève par l'appel à la routine de génération des fichiers de masques. Ils sont écrits dans l'ordre qui a été donné dans le chapitre précédent. Toutes les informations nécessaires sont connues : le nom des objectifs et leur nombre, et elles ne doivent plus être modifiées.
Les trois derniers fichiers-MPS (révisions et modification) présents sur le même support (BI-11), ne sont représentés que par leur entête, ce qui permet au MPS de se dresser un dictionnaire des noms de fichiers lors de son premier accès à BI. Ce dictionnaire restera invariant tout au long de l'exécution.

2. CALCUL DES OPTIMA

Il s'agit encore d'une étape sans dialogue. C'est la même routine qui effectue le positionnement des arguments OBJ et SCALE, puis qui garnit une colonne de la matrice de paiement. On distingue donc trois types d'appel à cette routine, qui sont identifiés par une valeur particulière de USER1 :

- initialisation : USER1 = 11

  Le premier objectif est placé dans les arguments

- itération : USER1 = 15

  Il y a d'abord lecture de la colonne correspondant à l'objectif qui vient d'être optimisé , puis changement des valeurs des arguments. Le nombre d'itérations est égal à NOBJ-1

- terminaison : USER1 = 19

  La dernière colonne de la matrice est garnie.

Nous prenons comme convention que toutes les valeurs qui concernent les objectifs, c'est-à-dire la matrice de paiement, les cibles, les compromis représentent toujours $f^i(x) = \sum_{j=1}^{n} C_j^i.x^j$, et cela quel que soit le facteur d'échelle qui a été donné pour une fonction.
D'autre part, le MPS, quand il donne la valeur correspondant à une ligne, donne la valeur de la variable logique associée. Pour les objectifs, on a donc $u^i + \sum_{j=1}^{n} C_j^i.x^j=0$ et la valeur fournie et celle de u^i, quel que soit le facteur d'échelle.

Lors de la lecture du fichier AI qui contient la solution, il y a inversion du signe de chacune des valeurs lues pour respecter les conventions.

Après cette phase, il y a calcul complet des coefficients de poids. Cette routine est évitée si AUTOSCALE = OFF, puisque les valeurs 1 ont déjà été garnies lors de la lecture du fichier ATOI. La formule était

$$\alpha^i = \left| \frac{P_i^i}{P_i^i - m^i} \right| \cdot \sqrt{\sum_{j=1}^{n} (C_j^i)^2}$$

où m^i était la plus grande valeur dans la ligne i de la matrice P. Ce choix est correct quand la fonction est à minimiser, mais pour une maximisation (i.e. SCALE (I) < 0), il faut prendre pour m^i le plus petit élément.
Globalement, on obtient $m^i = P_k^i = \max_{j=1, 2, \ldots, r} (P_j^i \star \text{sign}(\text{SCALE}(i))$

La valeur absolue se justifie pour garder α^i positif. De plus, le facteur d'échelle intervient dans les coefficients de la fonction économiques, comme il sera décrit dans le paragraphe suivant. La formule complète des coefficients de poids devient :

$$\alpha^i = \left| \frac{P_i^i}{P_i^i - m^i} \right| \sqrt{\sum_{j=1}^{n} \left(\frac{C_j^i}{R_i}\right)^2}$$

$$= \left| \frac{P_i^i}{P_i^i - m_i^i} \right| \frac{1}{|R_i|} \sqrt{\sum_{j=1}^{n} (C_j^i)^2}$$

Si RANGE = ON, ces poids seront de plus ramenés entre 0 et 1 :

$$\pi^i = \frac{\alpha^i}{\sum_{i=1}^{r} \alpha^i}$$

3. LES FICHIERS DE REVISION

Il y a d'abord l'étape de choix des niveaux de satisfaction qui est une phase de dialogue. Suivant les commandes de l'utilisateur, certaines cibles qui ont été positionnées à la valeur optimale de l'objectif, sont modifiées.

Au retour dans l'ACL, la base courante est écrite sur AI pour diminuer le nombre d'itérations après les révisions, puis l'interface est appelé pour générer ces deux fichiers. Il s'agit d'étendre ceux qui ont été déclarés en même temps que les masques. La suite des éléments à générer a déjà été décrite dans la première partie. Nous allons plutôt détailler la prise en charge de l'option SCALE.

En se référent à ORCHARD-HAYS [13] pp. 14-17, le scaling des variables logiques s'effectue comme suit :

- en entrée, diviser les coefficients des variables structurelles de la ligne par R_i

$$\bar{u}^i + \sum_{j=1}^{n} \frac{a_j^i}{R_i} \cdot x^j = \frac{b^i}{R_i}$$

- en sortie, faire $u^i = \bar{u}^i . R_i$ pour obtenir la vraie valeur de la variable logique.

Nous allons étendre ce principe pour les contraintes propres à l'interface.
La forme générale devient $\sum_{j=1}^{n} \frac{C_j^i}{R_i} . x^j + \alpha^i . y^{+i} - \alpha^i . y^{-i} = \frac{M^i}{R_i}$ $i=1, 2, \ldots, r$

On vérifie aisément que :

- les poids gardent la propriété d'invariance des variables d'écart;
- la signification de ces variables est également conservée
  $y^{-i} > 0$ si la cible est meilleure que la valeur de la fonction
  $y^{+i} > 0$ dans le cas contraire

4. EVALUATION DU COMPROMIS

Après la génération des fichiers pour l'interface, ont lieu les deux révisions qui sont suivies :

- de la création du fichier de travail
  SETUP SOURCE = GPSTEM : CONV : 1, AUTOSCALE
- de la recréation du DELIMIT set, détruit par SETUP
  DELIMIT NONE, RLIST = GPSTEM : MASK : OBJ/BI
- du positionnement des arguments
  SET OBJ = OBJ : NORME : L1, RHS = #3, SCALE = 1
- du chargement de la base sauvée
  LDBASIS SOURCE = GPSTEM : BASIS/AI
- du calcul de l'optimum
  CRASH
  PRIMAL

Ensuite a lieu l'évaluation du compromis. Elle s'effectue en trois étapes :

- évaluation par l'utilisateur :
  C'est l'étape de dialogue numéro 6. L'algorithme peut se terminer sur commande de l'utilisateur si le compromis atteint le satisfait, sinon, il veut chercher à l'améliorer en relançant le déroulement de l'algorithme
- vérification de la cohérence :
  Il s'agit simplement de tester y^{+i} , i = 1, 2, ...,r. Si un seul est positif, l'algorithme se termine après avoir imprimé la solution complète.

- choix de la méthode d'amélioration :

Le test porte maintenant sur y^{-i} , i = 1, 2, ...,r. S'ils sont tous nuls et si la dernière étape n'était pas l'étape d'amélioration globale (ceci pour éviter un passage inutile sur la paramétrisation), il y a retour dans l'ACL pour préparer cette étape, sinon, c'est la préparation de l'étape de perturbation.

Comme les conventions qui ont été prises respectent le sens attribué aux variables d'écart, il n'est pas nécessaire de faire intervenir les facteurs d'échelle dans ces tests.

5. MODIFICATIONS DES CIBLES

Soient les tableaux suivants :

RESULT (NMAX) : le compromis
LEVEL (NMAX) : les cibles
SCALE (NMAX) : les facteurs d'échelle
RANGE (2, NMAX) : les bornes de variation pour les cibles
SUBSTI (NMAX, NMAX): la matrice contenant les coûts de substitution entre les objectifs

L'étape de modification se découpe en quatre phases :

5.1. Lecture des valeurs

Cette phase se déroule avant le dialogue et est constituéede la lecture du fichier SO. Les problèmes de reconnaissance des valeurs ont déjà été abordés dans le paragraphe sur les fichiers.

Pour les bornes de variation, en lisant les valeurs, il faut effectuer leur descaling : RANGE(I,J) = RANGE(I,J) * SCALE(J) , I = 1, 2
De plus, si SCALE(J)$<$ 0, effectuer une permutation pour que

$$RANGE(1,J) \leqslant RANGE(2,J)$$

Pour les coûts de substitution :

La partie de la matrice mise à jour qui est donnée par INVOUT a la signification suivante : l'élément (i,j) indique la variation apportée à la variable logique de l'objectif i quand le second membre de la constrainte j varie d'une unité.

Dans un premier temps, cette matrice est textuellement copiée dans SUBSTI, puis on fait une double transformation pour :

- obtenir la variation sur la fonction économique ;
- tenir compte des facteurs d'échelle dans le second membre de j

SUBSTI(I,J) = - SUBSTI(I,J)/SCALE(J)

L'élément (i,j) signifie alors : la variation apportée à l'objectif i pour une modification unitaire de la cible de l'objectif j.

5.2. La sélection du niveau

Soit K, l'indice de l'objectif cité dans la commande RNGLVL :

- imprimer RANGE (1,K) et RANGE(2,K)
- mettre RELAX à zéro, où RELAX est la perturbation qui concerne le niveau qui a été introduit.

5.3. Le compromis estimé

La valeur de la commande MODIFY est placée dans RELAX :

- si RANGE(1,K) $\leq$ LEVEL(K) + RELAX $\leq$ RANGE(2,K), la solution imprimée sera exacte. Dans le cas contraire, envoyer un message d'avertissement;
- imprimer les valeurs du futur compromis

RESULT(I) + RELAX * SUBSTI (I,K) I = 1, 2, ..., NOBJ

Ces deux dernières commandes peuvent évidemment se répéter pour permettre l'exploration du voisinage du compromis.

5.4. La modification apportée

Lors de l'introduction de la commande EXECUTE, si le contexte est bien celui de cette étape, il y a appel à la routine qui génère un fichier de modification :

- effectuer le changement de niveau : LEVEL(K) = LEVEL(K) + RELAX
- modifier l'élément de la matrice dont :
  - la ligne est la contrainte numéro K, générée par l'interface
  - la colonne est < rhs > , le second membre de cette exécution
  - la nouvelle valeur est LEVEL(K)/SCALE(K)

6. AMELIORATION GLOBALE

6.1. Préparation du second membre composé

Il sera placé dans le vecteur COEFF (NMAX) qui représente les diminutions apportées aux seconds membres des contraintes générées par l'interface quand THETA vaut 1.

Si SAME = ON, faire COEFF(I) = -1 , I = 1, 2, ..., NOBJ, et positionner l'argument CRHS avec le nom et le KJ du vecteur fixe.

Sinon : - lire dans COEFF les bornes inférieures de variation, résultat du RNGRHS;

- faire COEFF(I) = COEFF(I) - LEVEL(I)/SCALE(I)
  I = 1, 2, ..., NOBJ
- générer le fichier de modification avec ce nouveau second membre;
- actualiser l'argument CRHS avec le nouveau nom et le nouveau KJ.

6.2. Test de la valeur de THETA

Après la paramétrisation, si THETA reste nul, il n'a pas d'amélioration globale possible et le retour se fait à l'étape de perturbation.

Sinon, : - changer la valeur des niveaux de satisfaction
  LEVEL(I) = LEVEL(I) + THETA* COEFF(I) * SCALE(I)
  I = 1, 2, ..., NOBJ
- générer les modifications correspondantes dans le MPS, pour les éléments dont les lignes sont les contraintes créées par l'interface
  La colonne est <rhs >, le second membre de cette exécution
  Les valeurs sont LEVEL(I)/SCALE(I) I = 1, 2, ..., NOBJ

o

o o

CONCLUSIONS

L'objectif de ce travail a été de mettre au point un outil opérationnel d'aide à la décision.

Ce souci a d'abord guidé l'élaboration de l'algorithme théorique. Pour mieux rencontrer la démarche des décideurs, le modèle adopté est multicritère et la technique de résolution a été voulue interactive.
De plus, nous avons insisté sur les niveaux de satisfaction ainsi que sur la facilité d'utilisation et la qualité des informations fournies.
D'autre part, "aide à la décision" n'implique pas, à notre sens, d'imposer une solution. C'est pourquoi, nous nous sommes attachés aussi à permettre l'exploration des solutions efficaces.

Nous avons ensuite pensé à utiliser la puissance de résolution d'un software de programmation linéaire, ce qui permettait de construire un outil réellement opérationnel. Nous n'avons pas rencontré de difficulté majeure pour trouver une solution ponctuelle aux différents problèmes posés par l'algorithme.

Il restait finalement à réaliser l'intrégration de toutes ces solutions. Nous avons donné les spécifications d'un langage de commandes pour contrôler le déroulement de l'interface, puis décrit son implémentation. Nous avons été amené à ajouter à l'algorithme des mécanismes tels que les points de reprise, pour améliorer son caractère opérationnel. La préoccupation qui a guidé cette étape était de réaliser un interface soigneusement intégré dans le package pour donner tout son sens au caractère interactif de l'algorithme.

Nous avons effectué différents tests qui portaient essentiellement sur le bon déroulement de toutes les fonctions. C'est pourquoi un premier prolongement du travail consisterait à tester l'interface sur des cas réels pour le confronter aux réactions des décideurs. Les problèmes qui seraient mis en évidence permettraient d'apporter de nouvelles améliorations.

Une autre direction de travail consisterait à implémenter l'algorithme et son langage sur d'autres softwares de programmation linéaire. Nous avons déjà mentionné les principaux points critiques à examiner pour un tel travail.

En dernier lieu, on peut songer à des extensions sur le MPS/66. Une première possibilité consisterait à résoudre des problèmes de programmation convexe en utilisant la programmation séparable.
Cependant, il ne faut pas négliger l'accroissement des temps de calcul qui en résulterait et qui nuirait au caractère interactif que nous avons recherché.
Il serait beaucoup plus intéressant de réaliser d'autres algorithmes sur le MPS/66 à partir des techniques de communication et de contrôle qui ont été mises en évidence pour cet interface. De la même manière, il serait possible de rendre certaines fonctions existantes beaucoup plus interactives.

°

° °

ANNEXE : EXEMPLES D'UTILISATION

1. MODES DE FONCTIONNEMENT

Nous décrivons rapidement les détails pratiques d'utilisation de l'interface.

1.1. Mode interactif

- La déclaration des fichiers 07 et 08 est superflue dans ce cas-ci.
- Pour se connecter avec l'interface à partir d'une console Time-Sharing, attendre que le job soit en exécution, ensuite
  . si l'UMC possède la permission TALK

```
*JDAC   GPSTEM
```

  . dans les autres cas

```
CTL   C
CTL   A
program name - GPSTEM
```

- Pour signaler qu'il est prêt à recevoir une commande, l'interface imprime une / en début de ligne.

1.2. Mode batch

- Les images-cartes des commandes doivent être contenues dans un fichier séquentiel permanent, en format BCD, sans numéro de ligne et cadrées à gauche.
- Sur le listing de sortie (file-code 08), les commandes sont recopiées à leur place respective, précédées également d'une /.

2. LE PROBLEME

L'exemple retenu pour illustrer les différentes fonctions est un tout petit problème de 8 variables, 8 contraintes et 5 objectifs appelés respectivement OBJ:1, OBJ:2, ..., OBJ:5. Le vecteur second membre s'appelle RHS et les données se trouvent dans GPSTEM:DATA sur IN.

Nous avons en outre testé l'interface sur deux autres matrices plus importantes : (68x117) et (607x990).

Si des modifications doivent être apportées pour certains exemples, elles seront données avec chaque cas particulier.

(1) Le fichier H★ contenant l'interface sous forme exécutable est déclaré avec deux files-codes : HS : pour permettre le chargement dynamique des routines par le MPS
H★ : contrainte imposée par le GCOS, car les routines ont une structure d'overlay.

(2) Les données MPS se trouvent sur IN.

(3) I★ contient l'ACL (voir paragraphe suivant).

(4) Les fichiers 07 et 08 simulent le terminal ; ils n'ont pas été utilisés puisque la connexion a été établie avec l'interface.

(5) Les différents fichiers de communication entre le MPS et l'interface sont temporaires, il n'y a pas de point de reprise.

```
$$     21200  H1700  CII-66 AT 21.206 FROM SYSTEM-0 T/S      1-
0001   $       SNUMB    21200
0002   $       COMMENT  CASA                    TSS CARDIN
0003   $$      USERID   CASA$#########
0004   $       IDENT    31CASA ,SCIE-ICDJL
0005   $$      USERID   CASA$#######
0006  A$       PROGRAM  MPS
0007   $       LIMITS   07,33K,,10
0008   $$      PRMFL    **,R,R,CASA/[illegible]
0009   $$      PRMFL    HS,R,R,CASA/JLL?/HSTAR
0010   $$      PRMFL    H*,R,R,CASA/JLL?/HSTAR
0011   $       FILE     ZS,Z1R,10R
0012   $       FILE     CC,Y4R,10R
0013   $       FILE     DD,X5R,10R
0014   $       FILE     EE,X6R,10R
0015   $       FILE     FF,X7R,10R
0016   $       DATA     IN
0017   $       DATA     I*
0018   $$      PRMFL    07,R,L,CASA/JLL?/C[illegible]
0019   $       SYSOUT   05
0020   $       SYSOUT   XO
0021   $       FILE     11,B1R,3L
0022   $       FILE     10,B1R
0023   $       FILE     01,B2R,1L
0024   $       FILE     11,B2R
0025   $       FILE     02,B4R,1L
0026   $       FILE     SO,B5R,10L
0027   $       FILE     12,B5R
0028   $       SYSOUT   14
0029   $       ENDJOB
  TOTAL CARD COUNT THIS JOB = 000034
```

Figure 1.

4. L'ACL

(1) Les macros nécessaires à l'interface sont rassemblées dans le fichier CASA/JLLP/MACRO. Elles sont insérées dans l'ACL par cette carte de contrôle.

(2) Appel de la macro GPSTEM :
- Le fichier-problème, résultat de CONVERT, s'appelle GPSTEM : CONV.
- Il est situé sur PT.
- Le second membre est RHS.
- On ne désire pas l'impression de la solution après chaque optimisation.
- On n'introduit pas de résultats antérieurs.

```
10        PREPRO
20$       SELECTA CASA/JLLP/MACRO                    (1)
30        CONVERT SOURCE=GPSTEM:DATA/IN,IDENT=GPSTEM:CONV
40        GPSTEM  GPSTEM:CONV;PT;RHS;NONE;NONE       (2)
50        EXECUTE
```

Figure 2.

5. EXEMPLES

5.1. Exemple général

(1) Après avoir lancé le job par CARDIN, attendre qu'il soit en exécution pour se connecter par

*JDAC GPSTEM

(2) Message indiquant l'étape de déclaration.

(3) Le modèle non pondéré est choisi et toutes les fonctions économiques interviennent pour cette exécution

```
2097C-01 - EXECUTING @ 21.052                  (1)
*JDAC GPSTEM
      ********************************************
      *  GPSTEM  INITIATED  ON  01/10/77  AT  21.058  *
      ********************************************

      DECLARATION PHASE                        (2)
/SET AUTOSCALE=OFF
/EXECUTE                                       (3)
```

Figure 3.a.

(4) Message indiquant l'étape de choix des niveaux de satisfaction ou "étape 3".

(5) Affichage de la matrice de paiement. Les cibles corespondent à l'optimum de chacune des fonctions.

(6) Message indiquant l'étape d'étude du compromis, ou "étape 6".

(7) L'utilisateur est satisfait de cette solution ; l'impression des états est supprimée et l'algorithme est clôturé.

(8) Message envoyé par GCOS pour indiquer la fin de l'exécution du programme. Il y a retour au système TSS existant avant la connexion à GPSTEM.

```
     SELECT YOUR SATISFACTORY LEVELS                    (4)
/DISPLAY PAYOFF

 DISPLAY OF PAYOFF MATRIX                               (5)

                                  OPTIMIZATION OF :
                          OBJ    :5      :            OBJ    :4      :
                          - - - - - - - - - - - - - - - - - - - - - - -
OBJ    :5      :                   -39.347518                -26.047619
OBJ    :4      :                    95.063830                -18.000000
OBJ    :3      :                  -179.06383                 -66.000000
OBJ    :2      :                  -170.55319                 -52.047619
OBJ    :1      :                   -8.5106384                -13.952381
                                  ====================================

                                  OPTIMIZATION OF :
                          OBJ    :3      :            OBJ    :2      :
                          - - - - - - - - - - - - - - - - - - - - - - -
OBJ    :5      :                   -39.347518                -38.611111
OBJ    :4      :                    95.063830                 92.333333
OBJ    :3      :                  -179.06383                 -176.33333
OBJ    :2      :                  -170.55319                 -176.83333
OBJ    :1      :                   -8.5106384                 0.50000000
                                  ====================================

                                  OPTIMIZATION OF :
                          OBJ    :1      :
                          - - - - - - - - - - - - - - - - - - - - - - -
OBJ    :5      :                   -27.000000
OBJ    :4      :                    5.5000000
OBJ    :3      :                   -89.500000
OBJ    :2      :                    27.750000
OBJ    :1      :                   -117.25000
                                  ====================================
/EXECUTE
```

FIGURE 3b

```
     WHAT DO YOU THINK ABOUT THIS COMPROMISE ?   (6)
/DISPLAY RESULT,SLACK

 DISPLAY OF CURRENT COMPROMISE SOLUTION
                              - - - - - - - - + - - - - - - - - - - -
 OBJ    :5      :                -39.347518
 OBJ    :4      :                 95.063830
 OBJ    :3      :               -179.06383
 OBJ    :2      :               -170.55319
 OBJ    :1      :                -8.5106384
                              ====================================

 DISPLAY OF SLACK VALUES (WEIGHED)
                              Y:PLUS                    Y:MINUS
                              - - - - - - - - - - - - - - - - - - - -
 OBJ    :5      :                 0.                       0.
 OBJ    :4      :                 0.                      113.06383
 OBJ    :3      :                 0.                       0.
 OBJ    :2      :                 0.                       6.2801399
 OBJ    :1      :                 0.                      108.73936
                              ====================================

/SET NOSO=ON      (7)
/ENDGP

                  (8)
ACTIVITY TERMINATED
```

FIGURE 3c

5.2. Modification et sauvetage-reprise à l'étape 6

(1) On a choisi un modèle pondéré. Affichage des coefficients de poids.

(2) Affichage de la solution actuelle.

(3) Impression sur XO de la solution complète (variables logiques et structurelles

(4) Sauvetage de l'état actuel avant toute modification, en vue d'une reprise éventuelle au cours de cette exécution. Il n'est donc pas nécessaire d'avoir des fichiers permanents.

(5) On effectue une première modification, qui ne figure pas dans l'exemple, pour amener la cible de OBJ:4 à la valeur -10.

(6) On voudrait une nouvelle modification du niveau de OBJ:4(valeur actuelle -10). Les valeurs fournies par MODIFY seront exactes si la nouvelle valeur est comprise entre -13.54 et -5.3641.

(7) La valeur dépasse les bornes, il y aurait changement de base.

(8) Une nouvelle modification est essayée. Elle respecte les bornes fournies, et on peut constater que les valeurs évaluées par l'interface correspondent bien aux valeurs effectivement calculées par le MPS.

(9) Reconfiguration du système à l'état précédant les deux modifications.

```
**********************************************
*  GPSIEM  INITIATED  ON  01/10/77  AT  21.435  *
**********************************************

     DECLARATION PHASE
/EXECUTE

     SELECT YOUR SATISFACTORY LEVELS
/DISPLAY WEIGTH                    (1)

 DISPLAY OF WEIGTH COEFFICIENTS
                              - - - - - - - - - - - - - - - - - -
 OBJ    :5      :                 8.3678522
 OBJ    :4      :                 0.84241836
 OBJ    :3      :                 21.189011
 OBJ    :2      :                 10.372301
 OBJ    :1      :                 11.823920
                              ====================================
/EXECUTE

     WHAT DO YOU THINK ABOUT THIS COMPROMISE ?
/DISPLAY RESULT,SLACK

 DISPLAY OF CURRENT COMPROMISE SOLUTION   (2)
                              - - - - - - - - - - - - - - - - - -
 OBJ    :5      :                -26.047619
 OBJ    :4      :                -18.000000
 OBJ    :3      :                -66.000000
 OBJ    :2      :                -52.047619
 OBJ    :1      :                -13.952381
                              ====================================

 DISPLAY OF SLACK VALUES (WEIGHED)
                             Y:PLUS                  Y:MINUS
                              - - - - - - - - - - - - - - - - - -
 OBJ    :5      :                0.                  13.299899
 OBJ    :4      :                0.                  0.
 OBJ    :3      :                0.                  113.06383
 OBJ    :2      :                0.                  124.78571
 OBJ    :1      :                0.                  103.29762
                              ====================================
/OUTPUT   (3)

     WHAT DO YOU THINK ABOUT THIS COMPROMISE ?
/SAVE     (4)

     WHAT DO YOU THINK ABOUT THIS COMPROMISE ?
/EXECUTE

     LOOK FOR A GOOD PERTURBATION   (5)
```

Figure 4.a

```
     LØØK FØR A GØØD PERTURBATION
/RNGLVL ØBJ:4                    (6)

     RANGES FØR SATISFACTØRY LEVEL ØF ØBJ    :4      :
 LØWER BØUND =   -13.540000          UPPER BØUND =   -5.3641000
/MØDIFY 5.                       (7)

 WARNING : THIS MØDIFICATIØN WILL REQUIRE A BASIS CHANGE

           (EXPECTED) SØLUTIØN AFTER THIS MØDIFICATIØN
                              - - - - - - - - - - - - - - - - - - -
 ØBJ    :5      :              -30.650519
 ØBJ    :4      :              -5.0000000
 ØBJ    :3      :              -79.000000
 ØBJ    :2      :              -119.43945
 ØBJ    :1      :               40.439447
                              ===================================
/MØDIFY 4.                       (8)

           (EXPECTED) SØLUTIØN AFTER THIS MØDIFICATIØN
                              - - - - - - - - - - - - - - - - - - -
 ØBJ    :5      :              -30.546713
 ØBJ    :4      :              -6.0000000
 ØBJ    :3      :              -78.000000
 ØBJ    :2      :              -113.55017
 ØBJ    :1      :               35.550174
                              ===================================
/EXECUTE

     WHAT DØ YØU THINK ABØUT THIS CØMPRØMISE ?
/DISPLAY RESULT

 DISPLAY ØF CURRENT CØMPRØMISE SØLUTIØN
                              - - - - - - - - - - - - - - - - - - -
 ØBJ    :5      :              -30.546713
 ØBJ    :4      :              -6.0000000
 ØBJ    :3      :              -78.000000
 ØBJ    :2      :              -113.55017
 ØBJ    :1      :               35.550173
                              ===================================
/RESTØRE                         (9)

 RESTØRE STEP 6 IN PRØGRESS

     WHAT DØ YØU THINK ABØUT THIS CØMPRØMISE ?
/DISPLAY LEVEL,RESULT

 DISPLAY ØF SATISFACTØRY LEVEL VALUES
                              - - - - - - - - - - - - - - - - - - -
 ØBJ    :5      :              -39.347518
 ØBJ    :4      :              -18.000000
 ØBJ    :3      :              -179.06383
 ØBJ    :2      :              -176.83333
 ØBJ    :1      :              -117.25000
                              ===================================

 DISPLAY ØF CURRENT CØMPRØMISE SØLUTIØN
                              - - - - - - - - - - - - - - - - - - -
 ØBJ    :5      :              -26.047619
 ØBJ    :4      :              -18.000000
 ØBJ    :3      :              -66.000000
 ØBJ    :2      :              -52.047619
 ØBJ    :1      :              -13.952381
                              ===================================
/
```

Figure 4.b

5.3. Amélioration globale

(1) On se limite à deux fonctions : en faire la déclaration explicite.

(2) Choix de niveaux de satisfaction particuliers.

(3) Le compromis est égal aux niveaux ; essai d'amélioration globale.

(4) Affichage du nouveau compromis.

```
*************************************************
*  GPSTEM  INITIATED  ON  01/10/77  AT  21.747  *
*************************************************

     DECLARATION PHASE
/DECLARE OBJ:4,OBJ:5                    (1)
/DISPLAY NAMES

 DISPLAY OF OBJECTIVES NAMES AND SCALES
                          - - - - - - - - - - - - - - - - - -
 OBJ    :4      :              1.0000000
 OBJ    :5      :              1.0000000
                         ====================================

/EXECUTE

     SELECT YOUR SATISFACTORY LEVELS
/SETLVL OBJ:4=-5.,OBJ:5=-30.            (2)
/DISPLAY LEVEL

 DISPLAY OF SATISFACTORY LEVEL VALUES
                          - - - - - - - - - - - - - - - - - -
 OBJ    :5      :             -30.000000
 OBJ    :4      :             -5.0000000
                         ====================================

/EXECUTE

     WHAT DO YOU THINK ABOUT THIS COMPROMISE ?
/DISPLAY RESULT

 DISPLAY OF CURRENT COMPROMISE SOLUTION
                          - - - - - - - - - - - - - - - - - -
 OBJ    :5      :             -30.000000
 OBJ    :4      :             -5.0000000
                         ====================================

/EXECUTE

     GLOBAL IMPROVEMENT OF THE SOLUTION IN PROGRESS...   (3)

     WHAT DO YOU THINK ABOUT THIS COMPROMISE ?
/DISPLAY RESULT

 DISPLAY OF CURRENT COMPROMISE SOLUTION                  (4)
                          - - - - - - - - - - - - - - - - - -
 OBJ    :5      :             -30.613531
 OBJ    :4      :             -8.5811788
                         ====================================

/SET NOSO=ON
/ENDGP

ACTIVITY TERMINATED
```

Figure 5.

5.4. Sauvetage-reprise avec fichiers permanents

L'utilisateur désire sauver l'état atteint après l'étape 3, puis lors d'une exécution ultérieure reprendre à partir de cette étape.

Le sauvetage

cartes-contrôles GCOS

```
$$     2042D ENTERED CTI-66 AT 20.433 FROM SYSTEM-C T/S      1

0001   $        SJUMP    20420
0002   $        COMMENT  CASA                TSS CARDIN
0003   $*       USERID   CASA$#########
0004   $        IDENT    31CASA ,SCIENCEJL
0005   $*       USERID   CASA$########
0006   $J       PROGRAM  PS
0007   $        LIMITS   C7,33K,,1UK
0008   $$       PRMFL    **,R,R,CASA/IPS/IPSTA2-0
0009   *$       PRMFL    US,R,R,CASA/JLLP/USTAR
0010   $$       PRMFL    U*,R,R,CASA/JLLP/USTAR
0011   $$       PRMFL    PT,R/W,R,CASA/JLLP/PT          (1)
0012   $        FILE     ZS,Z1R,100
0013   $        FILE     CC,X4R,100
0014   $        FILE     DD,X5R,100
0015   $        FILE     EE,X6R,100
0016   $        FILE     FF,X7R,10R
0017   $        DATA     I*
0018   $        DATA     I*
0019   $$       PRMFL    07,R,L,CASA/JLLP/COMOR
0020   $        SYSOUT   03
0021   $        SYSOUT   X3
0022   $        FILE     11,51R,7L
0023   $        FILE     10,51R
0024   $$       PRMFL    PI,R/W,L,CASA/JLLP/MPSJI
0025   $$       PRMFL    11,R/W,L,CASA/JLLP/MPSRI       (2)
0026   $$       PRMFL    09,R/W,R,CASA/JLLP/SAVE
0027   $        FILE     30,05R,10L
0028   $        FILE     12,05R
0029   $        SYSOUT   14
0030   $        ENDJOB
  TOTAL CARD COUNT THIS JOB = 000035
```

Figure 6.

(1) PT est assigné à un fichier random permanent, qui a été créé avant. Comme il ne contient aucune information, utiliser le file-code PT.

(2) Pour utiliser les possibilités de SAVE-RESTORE, assigner BI-11 et 09 à des fichiers permanents.

L'ACL est identique à celui des exemples précédents.

L'exécution :

```
2042D-01 - EXECUTING @ 20.491
*JOAC GPSTEM
      ********************************************
      *  GPSTEM  INITIATED  ON  01/11/77  AT  20.510  *
      ********************************************

      DECLARATION PHASE
/EXECUTE

      SELECT YOUR SATISFACTORY LEVELS
/SAVE                    (1)

      SELECT YOUR SATISFACTORY LEVELS
/ENDGP

ACTIVITY TERMINATED
```

Figure 7.

(1) A l'étape de choix des niveaux de satisfaction, on demande la sauvetage de l'état courant. Cette exécution aurait avantageusement pu être faite au batch.

La reprise

On voudrait garder le fichier problème tel qu'il est ; pour cela, lui assigner le file-code XP (lecture seulement).

Les cartes-contrôles GCOS sont les mêmes que pour le sauvetage, excepté :

```
$   PRMFL   XP, R, R, CASA/JLLP/PT
```

Dans l'ACL :

```
10          PREPRC
20    $     SELECT CASA/JLLP/MACRO
40          GPSTEM  GPSTEM:CONV:XP:GPS:NONE:NONE    (1)
50          EXECUTE
```

Figure 8.

(1) Le verbe CONVERT a été supprimé puisque le fichier-problème est déjà constitué.

Utilisation du file-code XP qui contient GPSTEM:CONV et GPSTEM:SAVE:3.

L'exécution :

```
     ************************************************
     *  GPSIEM  INITIATED  ON  01/11/77  AT  20.624  *
     ************************************************

     DECLARATION PHASE
/RESTORE                                        (1)

 RESTORE STEP 3 IN PROGRESS

     SELECT YOUR SATISFACTORY LEVELS
/DISPLAY LEVEL

 DISPLAY OF SATISFACTORY LEVEL VALUES
                          - - - - - - - - - - - - - - - - - -
 OBJ   :5      :            -39.347518
 OBJ   :4      :            -18.010000
 OBJ   :3      :            -179.06333
 OBJ   :2      :            -176.83333
 OBJ   :1      :            -117.25000
                          ====================================
/STATUS                                         (2)

          STATUS REPORT
          -------------
 CPU TIME USED :      9.399 SEC
 NUMBER OF OBJECTIVE FUNCTIONS : 5
 NUMBER OF LGL'S , SIR'S AND RHS'S :     22
 OPTIONS USED
     SCALING OF SLACK VARIABLES :YES
     WEIGHTS IN RANGE 0-1 :NO
     SUPPRESS PRINTING OF SS :NO
     SAME IMPROVEMENT FOR FUNCTIONS :NO
/EXECUTE

     WHAT DO YOU THINK ABOUT THIS COMPROMISE ?
```

Figure 9.

(1) A l'étape de déclaration, demander RESTORE. L'interface est positionné directement à l'étape correspondante.

(2) L'état courant est l'état atteint précédemment. L'exécution peut alors continuer normalement.

BIBLIOGRAPHIE

[1] S.M. BELENSON, K.C. KAPUR : *An algorithm for solving multicriterion linear programming problems with examples*, Operational Research Quartely, 24 (1973), pp. 65-77.

[2] R. BENAYOUN, J. de MONTGOLFIER, J. TERGNY, O. LARITCHEV : *Linear Programming with multiple objective functions : step method (STEM)*, Mathematical Programming, vol. 1 (1971), pp. 366-375.

[3] A. CHARNES, W.W. COOPER : *Management models and industrial applications of linear programming*, vol. 1, John Wiley, New-York (1971).

[4] L. DEGEHET : *L'approche multicritère dans le planning à moyen terme des entreprises manufacturées* mémoire de fin d'étude, FNDP, Namur (1973-1974).

[5] J.S. DIEUDONNE : *Eléments d'analyse*, Gauthiers-Villars, Paris (1972).

[6] J.S. DYER : *A time-sharing computer program for the solution to the multiple criteria problem*, Management Science, vol. 19, n° 12 (Aug. 1973), pp. 1379-1383.

[7] J. DYER : *An empirical investigation of a Man-Machine interactive approach to the solution of the multiple criteria problem*, in *Multiple criteria decision making*, édité par Cochrane et Zeleny, University of South-Carolina Press (1973), pp. 202-216.

[8] J. FICHEFET : *GPSTEM : an interactive multi-objective optimization method*, Colloquia Mathematica Societatis Jànos Bolyai, vol. 12, *Progress in Operations Research*, édité par A. Prékopa, North-Holland, Amsterdam (1976), pp. 317-332.

[9] A.M. GEOFFRION, J.S. DYER, A. FEINBERG, *An interactive approach for multi-criterion optimization, with an application to the operation of an academic department*, Management Science, vol. 19, n° 4 (Dec.1972), pp . 357-368.

[10] HONEYWELL-BULL, Series 60 (level 66) : *Mathematical Programming System (MPS) : Implementation Guide*, DD63.

[11] HONEYWELL-BULL, Series 60 (level 66) : *Mathematical Programming System (MPS) : Agenda Control Language Reference Manual*, DD60

[12] HONEYWELL-BULL, Series 60 (level 66) : *Mathematical Programming System (MPS) : Input/Output Manual*, DD64.

[13] W. ORCHARD-HAYS : *Advanced linear programming computing techniques*, Mc Graw-Hill Books Company, New-York (1968).

[14] B. ROY : *Vers une méthodologie générale d'aide à la décision*, Metra, Vol. XIV, n° 3(1975).

[15] M. SIMONNARD : *Programmation linéaire*, Vol. 1 et 2, Dunod, Paris (1972).

[16] J. WALLENIUS :*Comparative evaluation of some interactive approaches to multicriterion optimization*, Management Science, Vol. 21, n° 12 (Aug. 1975), pp. 1387-1347 .

[17] P.L. YU : *A class of solutions for group decision problems*, Management Science, Vol. 19, n° 8 (April 1973), pp. 936-946.

[18] P.L. YU : *Introduction to dominations structures in multicriteria decisions problems*, in *Multiple criteria decision making*, op cit. pp. 249-261.

[19] P.L. YU : *Cone university, cone extreme points, and nondominated solutions in decision problems with multiobjectives*, Journal of Optimization Theory and Applications, Vol. 14, n° 3 (1974), pp. 319-377.

[20] P.L. YU, M. ZELENY : *The set of all nondominated solutions in the linear case and a multicriteria simplex method*, Journal of Mathematical Analysis and Applications, Vol. 49, n° 2 (Febr. 1975), pp. 430-468.

[21] M. ZELENY : *Linear multiobjective programming*, Springer-Verlag (1974).

★

★ ★